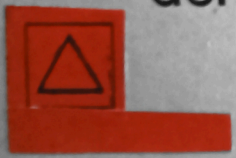
INFORMATIONSVERARBEITUNG

Heft 1

PROGRAMMIERUNG

der elektrischen Rechenanlage

C 8205 Z



Darf nicht i.f. Dibliographien angezeigt und nur mit Genehmigung der Militärbibliothek der DDR ausgeliehen werden

[Min. für Nationale Verteidigung]

INFORMATIONSVERARBEITUNG

Heft 1

Programmierung der elektrischen Rechenanlage C 8205 Z

Darf nicht in Bibliographien angezeigt und nur mit Genehmigung der Militärbibliothek der DDR ausgeliehen werden

Erarbeitet: Offiziersschüler R ü d i g e r , Michael

Inhaltsverzeichnis

		Seite
	Vorwort	4
1.	Konstruktion und Funktion des Kleinrechners C8205 Z	5
1.1.	Allgemeine Charakteristik	5
1.2.	Aufbau	6
1.3.	Speicher	6
1.4.	Steuerwerk und Rechenwerk	7
1.5.	Ein- und Ausgabegeräte	9
2.	Das Maschinenwort und seine Adressierung	10
2.1.	Das interne Maschinenwort	10
2.1.1.	Festkommazahlen	11
2.1.2.	Gleitkommazahlen	11
2.1.3.	Das Befehlswort	13
2.2.	Die externe Darstellung	14
2.3.	Die Adressierung	15
2.3.1.	Relative Adressierung	16
2.3.2.	Absolute Adressierung	17
2.4.	Wortkennzeichen	18
2.5.	Prinzipielle Regeln beim Programmieren	20
3.	Die wichtigsten Befehle	20
3.1.	Der absolute Sprung	20
3.2.	Der Lesebefehl	21
3.3.	Der Transport- und Löschbefehl	21
3.4.	Vergleicherbefehle	22
3.4.1.	Vergleichszustand	22
3.4.2.	Das reine Vergleichersetzen	23
3.4.3.	Die Vergleichersprungbefehle	24
3.4.4.	Der Negativsprung	25
3.4.5.	Der Positivsprung	25
3.5.	Der unbedingte Stop	26
3.6.	Der bedingte Stop	26
3.7.	Der Ausgabe-Halt-Befehl	27
3.7.1.	Die Anwahl eines E/A-Gerätes	27
3.7.2.	Die Ausgabe eines Zeichens	27
3.8.	Korrekturen im Programm	28
3.8.1.	Leer-Befehle	28
3.8.2.	Überspringen von Befehlen	29
3.8.3.	Einfügen von Befehlen	29
4.	Unterprogrammtechnik	30
4.1.	Das Anlegen von Unterprogrammen	31
4.2.	Programmieren mit Interpretiersystem	33
4.2.1.	Eingabe-Unterprogramme	33
4.2.2.	Ausgabe-Unterprogramme	35
4.2.3.	Gleitkomma-Rechenoperationen	36

Anlagenverzeichnis

		Seite
Anlage 1:	Programm "Extrapolation des Zielweges"	38
Anlage 2:	Zusammenstellung der wichtigsten Befehle	60
Anlage 3:	Zusammenstellung der wichtigsten Unterprogramme	61
Anlage 4:	Programmierungsvarianten für Vergleicherbefehle	63
Anlage 5:	Ausgabeadressen des R-300-Code und Anwahlbefehle für die E/A-Geräte	64
Anlage 6:	Literaturhinweise für tiefgreifendere Ausführungen über die Programmierung des C8205 Z	66
Anlage 7:	Interne Darstellung des Maschinenwortes	67
Anlage 8:	Abkürzungen und Symbole	68

Vorwort

Das vorliegende Studienmaterial ist ein Ergebnis der Arbeit des militärwissenschaftlichen Zirkels "Problemanalyse - Modellierung - Programmierung". Es dient der unmittelbaren Unterstützung der Ausbildung auf dem Gebiet der Mechanisierung und Automatisierung der Truppenführung und der Vermittlung von Grundkenntnissen im Ausbildungsfach "Informationsverarbeitung" in der Vierjahresausbildung.

Die Gestaltung des vorliegenden Studienmaterials sichert einen hohen Grad an Selbständigkeit und Selbsttätigkeit der Offiziersschüler beim Erwerb von Grundkenntnissen zur Programmierung des C8205Z.

Die Demonstration der Anwendung spezieller Befehle bzw. Befehlsgruppen durch praktische Beispiele und deren durchgängige Einordnung in das Programm "Extrapolation des Zielweges" ist eine zweckmäßige Methode zur Unterstützung einer problemhaften Gestaltung der Ausbildung.

Neben der Verwendung als Studienmaterial im Ausbildungsprozeß kann die vorliegende Arbeit auch als Nachschlagwerk für die unmittelbare Programmierung des Kleinrechners C8205Z im Rahmen von Beleg-, Diplom- und anderen militärwissenschaftlichen Arbeiten genutzt werden.

Oberstleutnant E. Bussenius
Fachgruppenleiter AFS

1. Konstruktion und Funktion der elektronischen Rechenanlage Carl 8205 Z

1.1. Allgemeine Charakteristik

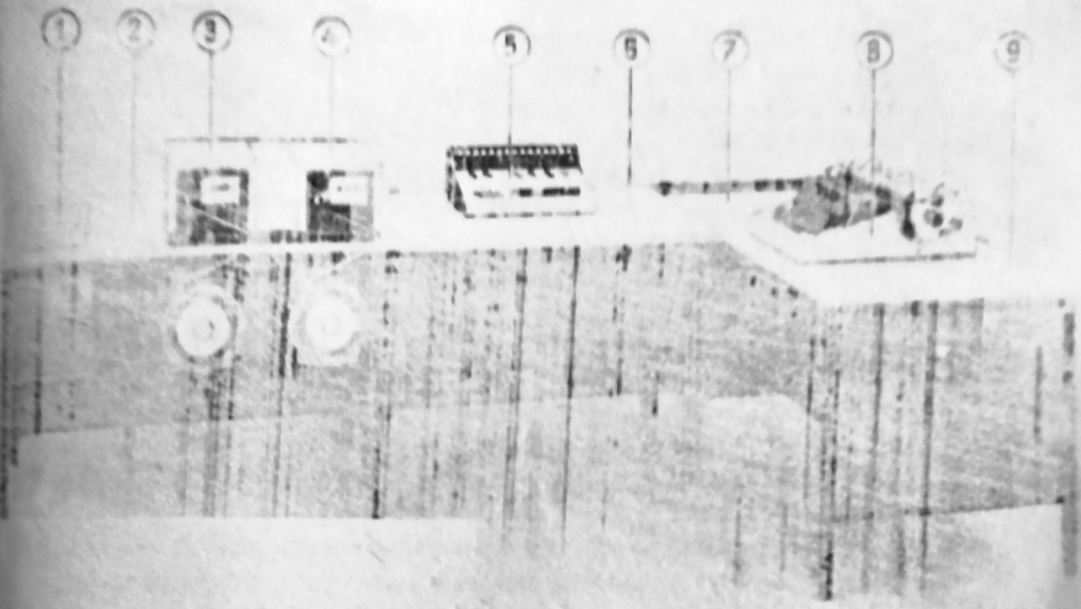


Abb. 1 Gesamtansicht

1 Lochbandstanzer
2 Peripherietisch
3 Leser 2
4 Leser 1
5 Tastatur-Anzeige-Einheit
6 Rechentisch
7 SVE-Tisch (Stromversorgungstisch)
8 Ein- und Ausgabe-Schreibwerk
9 Schreibwerktisch

Der Rechenautomat 8205 Z ist eine programmgesteuerte digitale elektronische Rechenanlage, die in Serie arbeitet. Der Rechner ist volltransistorisiert; die Zentraleinheit enthält ca. 400 Transistoren und 2000 Dioden. Dieser relativ geringe Aufwand wird durch Beschränkung auf ein Rechenregister (Akkumulator), mehrfache Ausnutzung von Baugruppen sowie durch Verzicht auf die Verdrahtung von Multiplikation und Division erreicht. Um trotzdem eine brauchbare Operationsgeschwindigkeit zu erzielen, wurden Befehlstypen, die besondere Eigenschaften des Trommelspeichers zum Rechnen ausnutzen, eingeführt. Bei optimaler Programmierung können ca. 2000 verdrahtete Operationen pro Sekunde ausgeführt werden.

Der 8205 Z ist eine Einadreß-Maschine. Als Hauptspeichereinheit wird

eine Magnettrommel mit einer Kapazität von 4096 Worten zu je 33 Bits verwendet. Die Arbeitsfrequenz des Rechners beträgt ca. 316 kHz; das entspricht einer Frequenz der Haupttrommel von ca. 18000 Umdrehungen pro Minute.
Das weitgehend analytische Befehlssystem ist außerordentlich flexibel. Mit nur etwa 9 verschiedenen Befehlstypen und etwa 23 Operationsarten lassen sich ca. 400 wirkungsmäßig unterschiedliche Befehle aufbauen. Durch die vielseitigen Ausrüstungsvarianten und Anschlußmöglichkeiten ist der 8205 Z sowohl für verschiedenste wissenschaftlich-technische als auch ökonomische Aufgaben zweckmäßig einsetzbar.

1.2. Aufbau

Die gesamte Rechenanlage besteht aus einer Tischkombination. Zur Zentraleinheit gehören die Recheneinheit, Tastatur und Anzeige. Die Grundvariante besitzt Eingabemöglichkeiten über die numerische Tastatur, das Schreibwerk 1 und ist weiterhin mit 2 Lochbandlesern ausgerüstet. Die Ausgabe erfolgt über einen Lochbandstanzer oder über das Schreibwerk 1.
Die periphere Ausstattung kann erweitert werden um maximal 4 Zusatztrommeln, ein 2. Schreibwerk oder Seriendrucker, 3. Lochbandleser, 2. Lochbandstanzer, einen Lochkartenleser und eine Kassetten-Magnetbandeinheit. Weiterhin sind Anschlußmöglichkeiten für das halbautomatische Datenerfassungssystem daro 1600 und frei belegbare Anschlußkanäle vorhanden.
Für die Lochbandeingabe im R 300-Code bzw. im Telegraphenalphabet Nr. 2 gibt es zwei hard-ware Varianten des Rechners. Jedoch sind auch andere Codes für die Lochbandein- und -ausgabe verwendbar.

1.3. Speicher

Die Magnettrommeln besitzen eine Kapazität von je 4096 Worten, die oktal von 0000 bis 7777 numeriert sind. Die Anordnung der Speicherplätze erfolgt in 128 Spuren in axialer Richtung mit je 32 Zellen radial. Die Hauptspeichertrommel wird auch Interntrommel IT genannt und hat eine Frequenz von 18000 Umdrehungen pro Minute. Damit ergeben sich Zugriffszeiten zwischen 0 und 3,33 ms. Die mittlere Zugriffszeit beträgt also 1,67 ms, die Wortzeit, d.h. die Zeit, in der sich die Interntrommel um eine Wortlänge (Zelle) weiterdreht, ca. 0,1 ms. Bei zeitoptimaler Programmierung entsprechen diese ca. 0,1 ms der Zugriffszeit.

Abb. 1: Magnettrommel

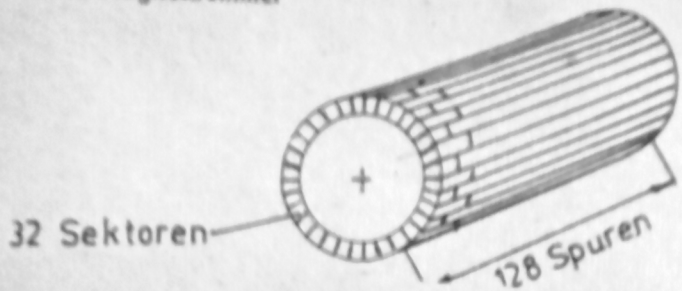


Abb. 1 Magnettrommel

Die 4 Zusatztrommeln (Externtrommeln ET) sind wahlweise anschließbar und haben eine Frequenz von ca. 3000 U/min; die durchschnittliche Übertragungsgeschwindigkeit einer Spur beträgt 60 ms. Die Übertragung ist in Blöcken mit jeweils 8 Wörtern von der Interntrommel zu einer der 4 Externtrommeln oder zurück möglich.
Die Entnahme von Informationen (bei der Interntrommel als Lesen bezeichnet) erfolgt zerstörungsfrei und ist beliebig oft möglich. Beim Abspeichern von Wörtern (bei der IT auch Transport genannt) wird die alte Information der Zelle automatisch gelöscht.

1.4. Steuerwerk und Rechenwerk

Bei der seriellen Arbeitsweise des 8205 Z werden die einzelnen Bits generell in der Reihenfolge vom niedrigsten zum höchsten übertragen und verarbeitet. Der 8205 Z ist ein Einadreßrechner; demzufolge werden die Befehle in der natürlichen Reihenfolge der Speicherzellen (d.h. entsprechend deren Numerierung) aufgerufen und abgearbeitet. Die Steuerung dieser Reihenfolge erfolgt über den Befehlszähler BZ, dessen Inhalt die Adresse des nächsten abzuarbeitenden Befehls darstellt.
Die Befehle selbst werden im Befehlsregister BR bereitgestellt. Nach der Abarbeitung des Befehls wird der Inhalt des Befehlszählers um eine Adresseneinheit automatisch erhöht (Abb. 2).
Intern arbeitet der Rechner dual, extern wird zur Vereinfachung soweit wie möglich die oktale Schreibweise verwendet.

Beispiel:
Wenn der Befehl Nummer 0627 abgearbeitet werden soll, muß sich die Adresse 0627 im Befehlszähler befinden. Danach wird der Inhalt der Zelle 0627 ebenfalls durch das Leitwerk vom Speicher in das Befehlsregister gebracht und die Adresse im Befehlszähler um 1 auf 0630 erhöht, so daß nach der Abarbeitung des Befehls 0627 als nächster 0630 folgt, usw.

Kurzschreibweise (vereinfacht):

BZ : = 0627

BR : = 0627 und

BZ : = 0630 (d.h. BZ : = BZ + 1 ^{0})

Durch Sprungbefehle kann die natürliche Reihenfolge der Befehlsabarbeitung durchbrochen werden; der Befehlszähler wird dabei neu eingestellt

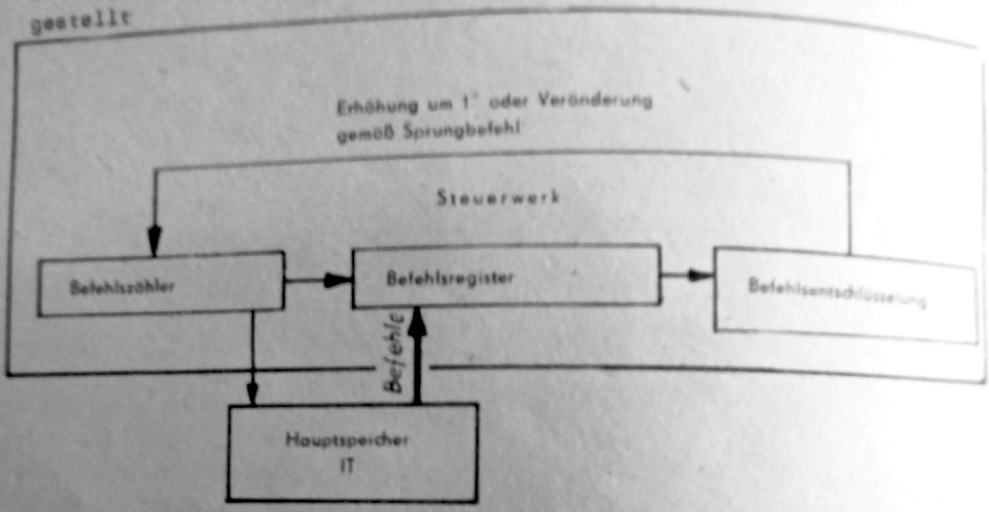


Abb. 2 Befehlsentnahme

Bei nichtorganisatorischen Befehlen erfolgt im Rechenwerk die entsprechende Operation. Dabei wird meist das im Akkumulator Ac stehende Wort mit dem Inhalt der Speicherzelle verknüpft, die durch den Adreßteil des Befehls (im Befehlsregister) angegeben ist und das Ergebnis in den Akkumulator oder bei einigen Befehlen auch in den Speicher gebracht (Abb. 3).

Befehlszähler und Akkumulator sind als Umlaufspeicher am Ende der Magnettrommel realisiert.

Die prinzipielle Anordnung eines Umlaufspeichers ist in Abb. 4 dargestellt.

Die einzelnen Bits des Wortes werden über den Sprechkopf S nacheinander aufgesprochen.

Durch die Drehung der Trommel unter den räumlichen feststehenden Köpfen wird die Information bis zum Hörkopf H mitgenommen. Dort wird sie abgehört, anschließend im Umformer U wieder zum Aufsprechen vorbereitet und, nachdem sie beim Sprechkopf S angelangt ist, wieder aufgesprochen. Das Wort hat also keinen festen Platz auf der Trommel, sondern läuft im beschriebenen Zyklus um. Wird dieser zwischen H und S unterbrochen, so wird durch den Löschkopf das Wort automa-

tisch gelöscht.

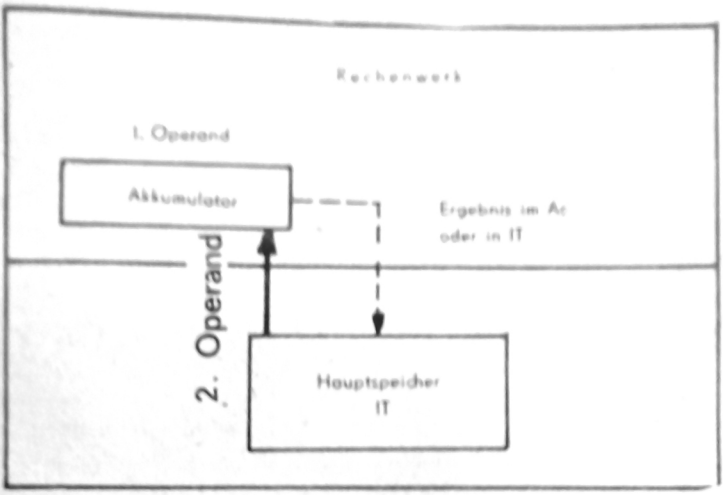


Abb. 3 Operationsdurchführung (vereinfacht)

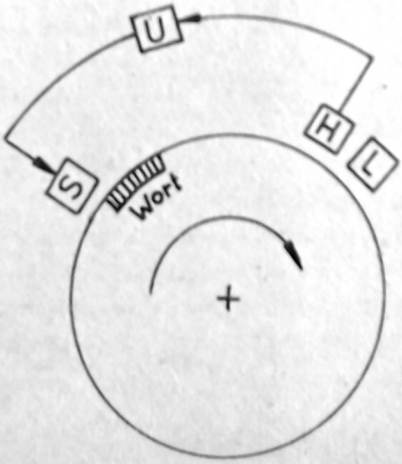


Abb. 4 Prinzip eines Umlaufspeichers (Ac und BZ)

Beim Umlaufspeicher wird zwar mehr Platz auf der Trommel benötigt als beim Hauptspeicher, dafür ergibt sich aber die Möglichkeit, die Information zu beliebigen Zeiten zu entnehmen oder aufzusprechen.

1.5. Ein- und Ausgabegeräte

Die Ein- und Ausgabe mittels Lochbandleser LBL und Lochbandstanzer LBS erfolgt 1-Zeichen-gepuffert (weitere Hinweise siehe Bedienungsanleitung, Heft 3).

Die Geschwindigkeiten sind programmabhängig und betragen:

LBL photoelektrisch, Typ 1210 = max. 200 Zeichen/Sekunde;
LBS = max. 50 Zeichen/Sekunde;
LKL = max. 200 Zeichen/Sekunde;
Schreibwerke = max. 10 Zeichen/Sekunde.

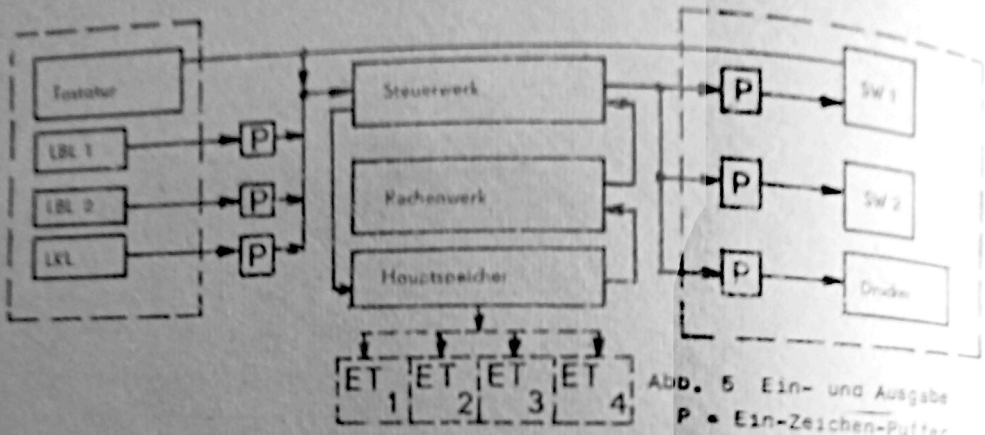


Abb. 5 Ein- und Ausgabe
P = Ein-Zeichen-Puffer

Bei der Eingabe gelangt das Zeichen vom Puffer in das Befehlsregister und bewirkt dort eine Veränderung des Adreßteiles und Umstellung des Eingabebefehls in einen Additions- oder Sprungbefehl, so daß dadurch eine dem Zeichen entsprechende Zahl vom Hauptspeicher entnommen und in den Akkumulator gebracht werden kann (ohne ein entsprechendes Programm kann keine Eingabeorganisation erfolgen).
Bei der Ausgabe kann durch den Ausgabebefehl nur ein bestimmtes Zeichen ausgegeben werden, das im Adreßteil des Befehls selbst direkt verschlüsselt und damit nicht direkt vom Speicher abhängig ist. Für die Ausgabe von Speicherzelleninhalten sind deshalb ebenfalls Programme notwendig.

2. Das Maschinenwort und seine Adressierung

2.1. Das interne Maschinenwort

Das Maschinenwort hat eine konstante Länge und besteht aus 33 Bit.
Es kann zur Darstellung von
- Zahlen;
- Befehlen;
- Alpha- und Sonderzeichen

dienen.
Die Art der Interpretation wird durch das Programm bestimmt, dabei kann ein Wort auch verschieden interpretiert werden.

Die Bits des Maschinenwortes werden mit 0 bis 32 von links nach rechts durchnumeriert; jedes Bit kann mit 0 oder 1 besetzt werden. Insgesamt gibt es somit unabhängig von der Interpretation 2^{33} verschiedene Wörter.
Jeweils ein Wort wird in einer Speicherzelle untergebracht.

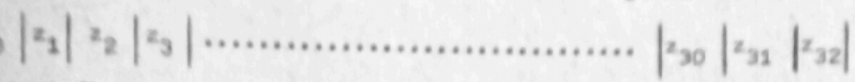


Abb. 6 Aufbau einer Speicherzelle

2.1.1. Festkommazahlen

Bei den verdrahteten Rechenoperationen (Addition und Subtraktion) wird das Maschinenwort als Dualzahl in Festkommadarstellung interpretiert. Das Bit z_0 wird dabei als Vorzeichen (Vz) interpretiert. Beim C8205 Z ist es möglich, eine Festkommadarstellung mit Kommainformation vorzunehmen, der Anschaulichkeit wegen wird im folgenden nur auf ganze Festkommazahlen eingegangen.
Bei positiven Zahlen (einschließlich der 0) ist $z_0 = 0$. Negative Zahlen werden als Zweierkomplement dargestellt. Sie sind durch $z_0 = 1$ charakterisiert. Bei negativen Zahlen werden alle Bits vor dem niedrigsten mit 1 besetzten negiert.
Es wird also bei z_{32} beginnend das erste Bit herausgesucht, welches ungleich 0 ist. Dieses Bit bleibt noch unverändert, alle anderen Bits werden negiert.

(1) Beispiel:

Bit	z_0	z_1	z_2	z_3	⟩ ⟨	z_{29}	z_{30}	z_{31}	z_{32}
x	0	1	0	1	⟩ ⟨	1	0	1	0
-x	1	0	1	0	⟩ ⟨	0	1	1	0

Zahlenbereich: $-2^{32} < z < 2^{32}$

Das Zweierkomplement ist eine eindeutige Darstellung.
Für die Stellenwertigkeit einer Festkommazahl ergibt sich:

z_0	z_1	z_2	z_3	z_4	⟩ ⟨	z_{30}	z_{31}	z_{32}
Vz	2^{31}	2^{30}	2^{29}	2^{28}	⟩ ⟨	2^2	2^1	2^0

Abb. 7 Stellenwertigkeit einer Festkommazahl

2.1.2. Gleitkommazahlen

Die Gleitkommazahl (GKZ) wird in halblogarithmischer Schreibweise als Produkt aus Mantisse m und Basis B mit dem Exponenten e dargestellt.

$$x = (-)\ m \times B^{(-)e}$$

(2) Beispiel mit Basis 10:

$$\begin{aligned} 456{,}789 &= 456{,}789 \times 10^{0} \\ &= 4{,}56789 \times 10^{2} \\ &= 0{,}456789 \times 10^{3} \\ &= 456789 \times 10^{-3} \end{aligned}$$

Da die Anzahl der Stellen vor dem Komma noch unterschiedlich ist, ist diese Darstellung für den Rechner noch nicht eindeutig, deshalb wird die GKZ "normiert".
Die Normierungsvorschrift besagt, daß die Mantisse der GKZ folgender Forderung entsprechen muß:

$$0{,}1 \leq m < 1$$

(3) Beispiel:

$$\begin{aligned} 456{,}789 &= 0{,}456789 \times 10^{3} \\ 0{,}00123 &= 0{,}123 \times 10^{-2} \\ -0{,}09561 &= 0{,}9561 \times 10^{-1} \end{aligned}$$

Beim Ausschreiben von GKZ durch ein Serviceprogramm werden folgende Vereinfachungen festgelegt:
- Die 0 vor dem Komma wird nicht geschrieben.
- Das Komma und die Basispotenz werden nicht geschrieben.

(4) Beispiel:

456,789 ⟶ 456 789 00 03
0,00123 ⟶ 123 000 00 -02
-0,09561 ⟶ -956 100 00 -01

Die interne Darstellung im Kleinrechner C8205 Z beruht auf der Basis 2.
Der Exponent darf bei der Eingabe nicht größer als 8 sein. Um zu vermeiden, daß ein negativer Exponent auftritt, wird er intern um 2^5 erhöht.

(5) Beispiel:

$$\begin{aligned} 10 &= 1010 \times 2^{0} \\ 0{,}06 &= 0{,}00001111 \ldots \times 2^{0} \\ -10 &= -1010 \times 2^{0} \\ -0{,}06 &= -0{,}00001111 \ldots \times 2^{0} \end{aligned}$$

Die normierte Darstellung sieht folgendermaßen aus:

$$\begin{aligned} 10 &= 0{,}1010 \times 2^{4} \\ 0{,}06 &= 0{,}1111 \ldots \times 2^{-4} \\ -10 &= -0{,}1010 \times 2^{4} \\ -0{,}06 &= -0{,}1111 \ldots \times 2^{-4} \end{aligned}$$

Daraus ergibt sich folgende interne Darstellung:

$10 = 0{,}1010 \times 2^{36}$

$0{,}06 = 0{,}1111 \times 2^{28}$

$-10 = -0{,}1010 \times 2^{36}$

$-0{,}06 = -0{,}1111 \times 2^{28}$

In der Triadendarstellung sehen diese Zahlen folgendermaßen aus:

10 = 2224 0000 000

0,06 = 1636 5605 075

-10 = 6224 0000 000

-0,06 = 5636 5605 075

z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7		z_{31}	z_{32}
Vz	e_5	e_4	e_3	e_2	e_1	e_0	m_1		m_{25}	m_{26}
	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}		2^{-25}	2^{-26}
	Exponent						Mantisse			

Vz - Vorzeichen der Mantisse

Abb. 8 Die interne Darstellung einer GKZ

Der interne Zahlenbereich ist durch

$$2^{-33} \leq x \leq (1 - 2^{-26}) \times 2^{31}$$

begrenzt.

2.1.3. Das Befehlswort

Die Darstellung eines Befehls erfolgt mit 21 Bit. Die ungenutzten Bit können für andere Informationen genutzt werden, z.B. für die Übermittlung von Parametern.

Das Befehlswort setzt sich folgendermaßen zusammen:

- Adreßteil (12 Bit);
- Operandenteil (9 Bit).

Diese 21 Bit werden zu Triaden zusammengefaßt (3 Bit = 1 Triade; 21 Bit = 7 Triaden).

Der Adreßteil setzt sich aus

- der Spurnummer (7 Bit) und
- der Sektornummer (5 Bit)

zusammen und dient zum Aufruf einer der 4096 Speicherzellen.

Der Operandenteil untergliedert sich in

- Befehlstyp B (3 Bit);
- Operationsart A (3 Bit);
- Operationszusatz Z (3 Bit).

Interne Daretellung eines Befehls siehe Anlage 7.
Somit ist theoretisch die Zusammenstellung von insgesamt $1000_{/8}$ Befehlen möglich, die jedoch nur 372 verschiedene Wirkungen haben.

2.2. Die externe Darstellung

Eine bitweise Notierung der Information außerhalb des Rechners, etwa zur Eingabe, wäre recht mühsam und umbequem. Deshalb wird je nach Art der Information eine besondere externe Darstellung verwendet. Auch die vom Rechner ausgegebenen Informationen erscheinen nach ihrer Art in einer von der internen Darstellung abweichenden Form. Alle externen Darstellungen sind von der Interpretation durch das Programm abhängig.
Unabhängig von der inhaltlichen Bedeutung der Information können im Wort (z.B. für die Eingabe) jeweils 3 Bit zu einer Oktalziffer oder Triade zusammengefaßt werden, so daß ein Maschinenwort 11 Triaden umfaßt.
Maschinenwort in Triadendarstellung siehe Anlage 7.
Diese Triadendarstellung ist die gebräuchlichste Form für Befehle. Auf sie wird auch bei der Erarbeitung der Befehlsliste zurückgegriffen.

C 8205 Z Befehlsfolge	Programm : a	Nr.: b
Bearbeiter: c	Datum: d	Blatt e von f

Sp. g	Nr. h	Kz. i	<AC> j	Bemerkungen <R_1> <R_2> <R_3> Schritt-Nr.
	0			
	1			
	2			
	3			

Abb. 9 Der Kopf der Befehlsliste

a - Bezeichnung des Programms,
b - Nummer des Programms,
c - Erarbeiter des Programms,
d - Datum der Programmerstellung,
e - Nummer des Blattes,
f - Gesamtzahl der Programmblätter, die das Programm umfaßt,
g - Spalte, in der Sprünge dargestellt werden können (innerhalb eines Blattes mit Pfeilen, über eine Seite hinaus mit Konnektoren entsprechend dem PAP),
h - Adresse des Befehls, wird durch das Vorschreiben von Oktalziffern ergänzt,
i - Kennzeichen des Befehls,
j - Parameter für den Befehl

k - Adresse des Speicherfeldes, mit der die Operation durchgeführt werden soll (Operand),
l - Operation,
m - Inhalt des Ac nach der Operation,
n - Inhalt der Register nach der Operation,
o - freie Spalte für Bemerkungen,
p - Schritt-Nummer laut Numerierung im PAP

Hinweis: - Die Spalten b, c, d brauchen nur auf dem ersten Blatt der Programmliste ausgefüllt werden.
- In den Spalten h, i, j, k, l dürfen in der Regel nur Oktalzahlen stehen.

Beim Erstellen des Programmlochbandes ist darauf zu achten, daß:
- nur die Spalten i, j, k, l abgelocht werden;
- die Zahlen fortlaufend ohne Leerzeichen zu lochen sind;
- jeder Befehl mit einem Punkt abgeschlossen wird;
- keine Sonderzeichen, wie z.B. die Lochung KB oder GB auf dem Lochband vorhanden sind.

Zur besseren Übersichtlichkeit ist es angebracht, jeden Befehl mit einem Punkt und anschließendem WRZL abzulochen.

2.3. Die Adressierung

Der C8205 verfügt über 4096 Speicherzellen auf seiner Interntrommel. Die Speicherplätze sind oktal von 0000 bis 7777 fortlaufend numeriert. Dabei geben die ersten 7 Dualzahlen die Spurnummer und die restlichen 5 Dualzahlen die Sektornummer an. Jedes Maschinenwort wird auf einer Speicherzelle abgespeichert und ist damit eindeutig adressierbar.
Programme werden fortlaufend ab einer bestimmten Adresse abgespeichert. Die Adresse des Befehls ist eine Oktalzahl. Bei der Eingabe wird diese Adresse bei jedem eingelesenen Befehl um eins erhöht. Jeder Befehl ist durch diese Adresse eindeutig adressierbar. Diese eindeutige Adressierbarkeit ist eine Voraussetzung für die Adresse im Befehl. Die Adresse im Befehl gibt an, mit welcher Speicherzelle die angegebene Operation durchgeführt werden soll. Die Adresse im Befehl kann sowohl relativ als auch absolut programmiert werden. Der Programmierer kann nicht über alle 4096 Speicherzellen der Interntrommel frei verfügen, da z.B. die Speicherplätze von 0000 bis 2777 und von 7500 bis 7777 oft mit Systemprogrammen und Bausteinen belegt sind. In der Regel ist dem Programmierer nicht vorher bekannt, welcher Speicherbereich für sein Programm frei ist, deshalb wird mit verschiebbaren Adressen, den relativen Adressen, gearbeitet.

2.3.1. Relative Adressierung

Bei der relativen Adressierung wird von der Adresse 0000 ausgegangen. Während der Eingabe wird der Adreßteil relativer Befehle um die Adresse erhöht, ab der die Abspeicherung des Programms beginnt.

(6) Beispiel:

relative Adresse	14
+ Programmbeginn	3000
absolute Adresse	3014

Die Adresse, ab der die Abspeicherung beginnt, bezeichnet man auch als Hauptleitadresse (HLA) oder 1. Leitadresse (LA).
Sie erfüllen die Funktionen:
- Bestimmung des Abspeicheranfangs;
- Adressenänderungen 1. Art, die zum Ansprung von Unterprogrammen und zur Parameterversorgung benötigt werden. Für die Adressenänderung 1. Art stehen 8 Nebenleitadressen zur Verfügung.

Jeder Befehl beginnt mit einem zweistelligen Kennzeichen, das die jeweilige Leitadresse, die zur relativen Adresse addiert wird, bestimmt:

01 - Zur relativen Adresse wird die 1. Leitadresse (Hauptleitadresse) addiert.
02 - Zur relativen Adresse wird die 2. Leitadresse (1. Nebenleitadresse) addiert.
.
.
.
09 - Zur relativen Adresse wird die 9. Leitadresse (9. Nebenleitadresse) addiert.

(7) Beispiel:

rel. Adresse des Befehls	Kennzeichen	Operand	Operation
121	01	124	000.

Das Programm beginnt ab der Adresse 3000. Nach dem Einlesen steht folgender Befehl auf dem Speicherplatz 3121 (rel. Adresse des Befehls (121) + Adresse Programmbeginn (3000) = Speicherplatz (3121)):

Speicherplatz	Operand	Operation
3121	3124	000.

Wenn dem gleichen Programm als 1. Nebenleitadresse 7000 zugewiesen wurde, so wird aus dem Befehl:

rel. Adresse des Befehls	Kennzeichen	Operand	Operation
233	02	0	025

Nach dem Einlesen folgender Befehl, der auf dem Speicherplatz 3233 steht (zur relativen Adresse des Befehls wird immer die 1. Leitadresse addiert, welche Adresse zum Operanden addiert wird, richtet sich nach dem Kennzeichen):

Speicherplatz	Operand	Operation
3233	7000	025

2.3.2. Absolute Adressierung

Die absolute Adressierung ganzer Programme wird kaum angewandt, teilweise ist die absolute Adressierung aber notwendig, so z.B.

- zum Ansprung von BIS-Bausteinen bzw. BIS-Unterprogrammen (Eingabe, Ausgabe, arithmetische Rechenoperationen mit GKZ);
- zur Adressierung von Versorgungszellen, Registern u.ä.;
- zur Ein-/Ausgabe von alpha-numerischen Zeichen;
- zur Auswahl eines Ein-/Ausgabe-Gerätes;
- zur Definition von Konstanten.

Eine absolut eingelesene Adresse bleibt in der programmierten Form erhalten, sie erfährt keine Veränderung mehr.
Absolut adressierte Operanden beginnen immer mit dem Kennzeichen 00 - der eingelesene Operand wird nicht verändert.
Mit der alleinigen Abspeicherung von 00. wird der entsprechende Speicherplatz mit 0 belegt.

(8) Beispiel: In einem relativ programmierten Programm wird eine Versorgungszelle 7745 für die nachfolgende Eingabe mit einem Abspeicherbefehl gefüllt. Das Programm beginnt auf dem Speicherplatz 3000.

rel. Adresse des Befehls	Kennzeichen	Operand	Operation
116	00	7745	240.

Nach dem Einlesen des Programms steht auf dem Speicherplatz 3116 folgende Anweisung:

Speicherzelle	Operand	Operation
3116	7745	240.

2.4. Wortkennzeichen

Wie schon im Punkt 2.3. erwähnt, wird das Wortkennzeichen (siehe 2.2., Spalte i) zur Kennzeichnung der relativen bzw. absoluten Adressierung der Befehle angewendet. Jeder Befehl muß ein Wortkennzeichen zugeordnet bekommen.
Das Wortkennzeichen erfüllt aber auch noch eine weitere Aufgabe, die für das Anlegen von Konstanten- und Datenfeldern von Bedeutung ist. Das Wortkennzeichen löst den Baustein aus, der entweder die Adreßrechnung oder die Konvertierung realisiert.
In der Regel wird am Anfang des Programms ein Bereich angelegt, der für die Abspeicherung der Daten und Konstanten dient. Die Reservierung der Datenfelder und die Definition der Konstanten wird an Hand des PAP vorgenommen. Deshalb ist es unbedingt erforderlich, daß man sich vorher darüber klar wird, welche Datenfelder und Konstanten man während des Programms benötigt, auch die Anzahl der Speicherzellen für Ergebnisse und für die Zwischenspeicherung muß bekannt sein.
Für alle einzulesenden Zahlen sind Speicherplatzreservierungen vorzunehmen. Im allgemeinen geht man davon aus, daß diese Speicherplätze als erstes durch die eingelesenen Daten belegt werden, so daß man sie im Programm mit Null definiert (Definition von Speicherplätzen mit 0 siehe 2.3.2.).

(9) Beispiel: Für die einzulesenden Koordinaten x_1, y_1 und z_1 sind fortlaufende Speicherplätze zu reservieren.
Die Speicherplätze sind mit 0 zu definieren.

000					
001	00				X_1
002	00				Y_1
003	00				Z_1
004					

Da in der Regel in der Ausbildung mit GKZ gerechnet wird, müssen alle Konstanten für das Programm als GKZ definiert werden.
GKZ werden definiert, indem man das einstellige Kennzeichen 3 vor die Zahl schreibt.

▷ Grundform: 3 (-) z.

In der Zahl z kann dabei eine Kommainformation enthalten sein, die Bedingungen, die bei der Arbeit mit GKZ beachtet werden müssen, werden im Punkt 2.1.2. behandelt.

(10) Beispiel: Für das Programm "Extrapolation des Zielweges" werden die Konstanten 1; 10; 0 und 0,06 als GKZ benötigt.

1	
22	31 ≙1"
23	31,0 ≙10"
24	30,,06 ≙0.06"
25	00 ≙0"
6	

Hinweis: Die interne Darstellung der Null ist bei allen angegebenen Wortinterpretationen gleich, Die Definition

4	
25	30 ≙0"
6	

hat also die gleiche Wirkung wie die Definition

4	
25	00 ≙0"
6	

Im allgemeinen wird die 2. Definition verwendet.

Trade	t_{10}			t_9			t_8			t_7			t_6			t_5			t_4			t_3			t_2			t_1			t_0		
Bit	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8	Z_9	Z_{10}	Z_{11}	Z_{12}	Z_{13}	Z_{14}	Z_{15}	Z_{16}	Z_{17}	Z_{18}	Z_{19}	Z_{20}	Z_{21}	Z_{22}	Z_{23}	Z_{24}	Z_{25}	Z_{26}	Z_{27}	Z_{28}	Z_{29}	Z_{30}	Z_{31}	Z_{32}
Duale	0	1	1	1	0	1	1	0	0	0	0	0	1	1	1	0	0	1	1	1	0	1	0	1	0	1	0	1	1	0	0	1	1
Oktal	3			5			4			0			7			1			6			5			2			6			3		

Abb. 10 Übersicht über die Wortkennzeichen

00 - absolute Programmierung, der Befehl wird unverändert abgespeichert,
01 - relative Adressierung, zum Operanden des Befehls wird die HLA addiert,
02 - relative Programmierung, zum Operanden wird die 2.
09 (... bzw. 9), - Leitadresse addiert,
3 - Eingabe einer Dezimalzahl (DZ) (interne Konvertierung in GKZ),
4 - Eingabe einer GKZ,
5 - Eingabe einer DZ (int. Konv. in FKZ),
6 - Eingabe einer DZ (int. Konv. in eine Zahl in Tetradendarstellung),
7 - Eingabe einer DZ (int. Konv. in FKZ),
8 - Eingabe von DZ ohne Kennzeichen im Satz (int. Konv. in GKZ),
9 - Eingabe von DZ ohne Kennzeichen im Satz (int. Konv. in GKZ)

2.5. Prinzipielle Regeln beim Programmieren

- Parameter- und Konstantenfelder sind zu umspringen.
- Vor Ein-/Ausgaben ist grundsätzlich die Kanalanwahl durchzuführen.
- Führende Nullen im Adreßteil brauchen nicht auf das Programmformular geschrieben und abgelocht werden (Adresse 0320 ⟶ 320).
- In den Spalten <Ac> und Bemerkungen des Programmformulars sollten alle Zeitzustände notiert werden.
- Ein detaillierter PAP erleichtert die Programmierung.
- Die verwendeten Leitadressen sind als Notiz für die zu erarbeitende Regieanweisung festzuhalten.
- Die Speicherplätze sind fortlaufend zu besetzen, Lücken im Programm sind nicht statthaft.
- Als erstes sollte man sich an Hand des PAP eine Zusammenstellung aller Konstanten und Variablen erarbeiten.
- Der prinzipielle Befehlsaufbau: Kz (max. 2-stellig) - Operand (max. 4-stellig) - Operation (3-stellig) ist einzuhalten.
- Zur besseren Übersichtlichkeit ist nach Möglichkeit nur ein Bereich für Konstantendefinitionen und Speicherplatzreservierungen anzulegen.
- Alle Adressen sind als Oktalzahlen zu programmieren.

3. Die wichtigsten Befehle

3.1. Der unbedingte Sprung

Durch den unbedingten Sprung wird die natürliche Reihenfolge der Befehlsabarbeitung verlassen, die Abarbeitung wird an der durch den Adreßteil des Befehls bestimmten Stelle fortgesetzt.

▷ Grundform: a 000.

(11) Beispiel: Von Speicherplatz 1 bis 32 sind Konstanten und Variablen-Felder angelegt worden. Die Ansprungadresse des Programms ist 0. Deshalb muß auf dem Speicherplatz 0 ein Befehl stehen, der das Überspringen der nachfolgenden $32_{/8}$ Speicherplätze sichert.

000	01	33 000	
001		⋮	

Im allgemeinen fängt jedes Programm mit einem unbedingten Sprungbefehl an, denn die nachfolgenden Felder werden für Speicherplatzreservierungen genutzt.

3.2. Der Lesebefehl

Der Lesebefehl gewährleistet das Löschen des Akkumulators (Ac) und das nachfolgende Einlesen des Inhaltes der Speicherzelle, die im Adreßteil des Befehls bestimmt wird.

$$<Ac> := <a>$$

▷ Grundform: a 211.

Der Inhalt der Speicherzelle wird bei dieser Operation nicht verändert.

(12) Beispiel: Für die nachfolgende Subtraktion wird der Wert x_1 zuerst in den Ac gelesen.

204		⋮	
205	01	1,211	$<Ac> := x_1$
206		⋮	

001 - x_1

3.3. Der Transport- und Löschbefehl

Der Transportbefehl transportiert den Inhalt des Ac zu der im Adreßteil des Befehls angegebenen Speicherzelle, der alte Inhalt der Speicherzelle wird dabei überschrieben, der Inhalt des Ac bleibt erhalten.

$$<a> := <Ac>$$

▷ Grundform: a 240.

(13) Beispiel: Der Wert x_1, der im Beispiel 12 in den Ac gelesen wurde, soll nun in das Register 1 (>7773<) transportiert werden.

204	, , ⋮ ,	
205	01, , 1, 211	<Ac> := x1
206	00, ,7773, 240	<R1> := x1
207	, , ⋮ ,	

Durch Hinzufügen der Löschanweisung zum Transportbefehl erhält man den Löschbefehl. Dieser Befehl löscht zuerst den Ac und transportiert anschließend die Null auf den im Operandenteil des Befehls angegebenen Speicherplatz.

$$<a> := <Ac> := 0$$

Grundform: a 241.

(14) Beispiel: Zur Demonstration dieses Befehls werden die Speicherplätze 14 und 15 auf Null gelöscht.

202	⋮	
203	01 14 241	Vy := 0
204	01 15 241	Vz := 0
205	⋮	

Um Speicherplätze auf Null zu löschen, kann man natürlich auch zuerst eine Null in den Ac lesen und diese dann anschließend auf die Speicherplätze transportieren.

177	, , : ,	
200	01, , 25, 211	<Ac> := 0
201	01, , 12, 240	V := 0
202	01, , 13, 240	Vx := 0
203	, , : ,	

025 - "0"

3.4. Vergleicherbefehle

3.4.1. Der Vergleichszustand

Grundsätzlich unterscheidet man bei der Arbeit mit Vergleicherbefehlen 2 Etappen: das Einstellen des Vergleichers durch Vergleichersetzbefehle und das Auswerten dieses Vergleichszustandes durch entsprechende bedingte Sprungbefehle. Beide Etappen müssen nicht unbedingt aufeinander folgen, zwischen ihnen dürfen jedoch keine Bausteine oder Unterprogramme angesprungen werden, da diese in der Regel mit Vergleichern arbeiten.

Der C8205 Z ist mit einem Flip-Flop-Paar zur Speicherung von 3 möglichen Vergleichszuständen ausgerüstet, die durch 2 Kontrollampen

(V1, V2) angezeigt werden. Für die Darstellung wird der Inhalt der beiden Vergleicher-Flip-Flops als 2-stellige Dualzahl interpretiert.

V2	V1	Bedeutung			
0	1	Vergleicher 1 gesetzt	<Ac>	<	<a*>
1	0	Vergleicher 2 gesetzt	<Ac>	>	<a*>
1	1	Vergleicher 3 gesetzt	<Ac>	=	<a*>

Abb. 11 Mögliche Vergleichszustände

Diese Vergleicher sind für FKZ und GKZ anwendbar, es ist aber darauf zu achten, daß die zu vergleichenden Zeichen das gleiche Format haben.

3.4.2. Das reine Vergleichersetzen

Der Befehl Vergleichersetzen vergleicht den Inhalt des Ac mit dem Inhalt des im Operandenteil des Befehls angegebenen Speicherfeldes und setzt entsprechend die Vergleicher-Flip-Flops.

$$<Ac> \lessgtr <a^*> \ ?$$

▷ Grundform: a* 330.

(15) Beispiel: Ein Zyklus ist n_{max}-mal abzuarbeiten. Der Zähler wird additiv erhöht und anschließend wird er mit dem Sollwert n_{max} verglichen. Die Phase des Einstellens des Vergleichers umfaßt die Befehle:

353		⋮		
354	01	17	211	<Ac>:=Zähler
355	01	23	330	Zähler $\gtrless$ nmax?
356		⋮		

017 - Zähler
023 - n_{max} (10)

Eine Sonderform des Befehls Vergleichersetzen ist der Vergleich des Inhaltes eines Speicherplatzes mit 0. Bei diesem Befehl wird zuerst der Ac auf Null gelöscht und anschließend wird der Ac mit dem Inhalt des Speicherfeldes verglichen. Ein Einlesen des zu vergleichenden Maschinenwortes in den Ac ist bei diesem Befehl nicht notwendig.

$$<a^*> \lessgtr 0 \ ?$$

▷ Grundform: a* 331.

(16) Beispiel: Wenn beim Einlesen des LB das Zeichen "Letzter Datensatz" erkannt wird, soll ein Schalter, der sich auf dem Speicherfeld 20 befindet, auf eins gesetzt

werden. Nach Abarbeitung des Programms wird gefragt, ob der eingelesene Datensatz der letzte war, wenn es an dem ist, wird das Programm beendet. Das Einstellen des Vergleichers umfaßt dabei folgende Befehle:

460	⋮	
461	01 20 331	Schalter ≧ 0 ?
2	⋮	

3.4.3. Die Vergleichersprungbefehle

Durch die Vergleichersprungbefehle wird der im Rechner vorhandene Vergleichszustand ausgewertet und die Bedingungen für den Sprung nach der im Adreßteil des Befehls bestimmten Adresse festgelegt. Di Auswertung kann beliebig oft programmiert werden, wenn der abzufragende Vergleichszustand nicht durch andere Vergleichersetzbefehle verändert wird.

▷ Grundform: a 140.
a 150.
a 160.
a 170.

Programmierungsvarianten für Vergleicherbefehle siehe Anlage 4.

(17) Beispiel: Die Auswertung des Vergleichszustandes aus dem Beispiel 15 erfolgt mit den Befehlen:

353	⋮	
354	01, 17, 211	<Aç> :=Zähler
355	01, 23, 330	Zähler ≧ n max
356	01, 271, 160	Zähler ≦ n max→
357	⋮	Zykius- Beginn

Die Auswertung des Vergleichersetzens mit dem Befehl 331 (siehe Beispiel 16) erfolgt auf gleiche Weise:

460		
461	01, 20, 331	<Schalter> ≧ 0 ?
462	01, 115, 170	Gleichheit→nächste Eingabe
463	01, 474, 000	Ungleichheit → Sprung zu Ende
4		

3.4.4. Der Negativsprung

Der Negativsprung bewirkt ein Vergleichen des Inhaltes des Ac mit Null. Ist der Ac-Inhalt negativ, so wird ein Sprung zu der im Operandenteil des Befehls angegebenen Adresse ausgeführt, anderenfalls wird das Programm mit dem Abarbeiten des nächsten Befehls fortgesetzt. Vorheriges Vergleichersetzen und zusätzliche Vergleichersprungbefehle sind nicht notwendig.

$$<Ac> < 0 ?$$

▷ Grundform: a 040.

(18) Beispiel: Bei der Extrapolation der Höhenkoordinate wird ständig verglichen, ob der errechnete Wert kleiner als Null ist. Wenn der Höhenwert kleiner als Null ist, soll eine Fehlerausschrift ausgeschrieben werden, ansonsten wird die Abarbeitung normal fortgesetzt. Der errechnete Höhenwert steht bereits im Ac.

326	⋮	
327	01, 464,040	h<0⇒Fehleraus
330	⋮	schrift

3.4.5. Der Positivsprung

Der Positivsprung arbeitet nach dem gleichen Prinzip wie der Negativsprung. Er unterscheidet sich lediglich dadurch, daß er zu der im Operandenteil des Befehls angegebenen Adresse springt, wenn der Inhalt des Ac größer oder gleich null ist, anderenfalls wird das Programm mit dem nächsten Befehl fortgesetzt.

$$<Ac> \geq 0 ?$$

▷ Grundform: a 050.

(19) Beispiel: Von einer Zahl n wird bei jeder Abarbeitung eines Zyklusses eine eins subtrahiert. Das Programm beendet den Zyklus, wenn die Zahl negativ ist. (Diese Programmschritte sind nicht im Programm "Extrapolation des Zielweges" enthalten.)

557	⋮	
560	01, 23,211	<Ac>:=n
561	01, 320,050	n=0⇒Zykl-Be-
2	⋮	ginn

023 - n (10)

Programmierungsvarianten für die Befehle 040 und 050 siehe Anlage 4.

3.5. Der unbedingte Stop

Zur Unterbrechung des Programmablaufs an einer bestimmten Programmstelle ist ein Stopbefehl notwendig, anderenfalls würde der Rechner nach der letzten Operation weiterarbeiten und das nächste Maschinenwort als nächsten Befehl interpretieren.

▷ Grundform: a 020.

Der Operand hat bei der Ausführung dieser Operation keine Auswirkung.

(20) Beispiel: Das Programm soll nach Berechnung und Ausgabe der letzten Werte abgebrochen werden.

547		
550	01, , ,020	Stop

3.6. Der bedingte Stop

Der bedingte Stop wird dort angewendet, wo ein Programm an einer bestimmten Stelle zeitweise unterbrochen werden soll oder wo es der Entscheidung des Bedieners überlassen werden soll, ob er das Programm unterbricht oder nicht. Ein Beispiel für die Anwendung dieses Befehls wäre z.B. ein zeitweiser Halt zwecks eventuellem Einspannen eines neuen Blattes in das Schreibwerk.
Der Stop hängt von der Stellung des Schalters [BS] ab. Ist dieser Schalter zum Zeitpunkt der Abarbeitung des Befehls in der unteren Stellung eingerastet, wird der Ablauf wie beim unbedingten Stop unterbrochen. An der Konsole leuchtet die Lampe BS auf. Durch Lösen des Schalters BS oder durch Betätigung der Taste Start [ST] wird das Programm mit dem nächsten Befehl fortgesetzt. Ist der Schalter nicht in der unteren Stellung eingerastet, wird die Abarbeitung des Programms sofort mit dem nächsten Befehl fortgesetzt.

▷ Grundform: a 060.

Der Operand hat bei dieser Operation keinen Einfluß auf deren Ausführung.

(21) Beispiel: Bevor ein neuer Tabellenkopf ausgegeben wird, soll der Rechner das Programm zeitweise unterbrechen, um dem Bediener die Möglichkeit zu geben, eventuell einen Blattwechsel beim Schreibwerk durchzuführen.

55_1		
55_2	01, , ,060	BS
55_3		

3.7. Der Ausgabe-Halt-Befehl

Mit dem Ausgabe-Halt-Befehl können 2 Funktionen ausgeführt werden:
1. Anwahl eines E/A-Gerätes;
2. Ausgabe eines Zeichens.
Der Ac-Inhalt ändert sich durch diesen Befehl nicht und hat keinen Einfluß auf dessen Ausführung.

3.7.1. Die Anwahl eines E/A-Gerätes

Vor jeder Ein- bzw. Ausgabe macht es sich erforderlich, das E/A-Gerät, mit dem die Operation ausgeführt wird, anzuwählen. Die Anwahl bleibt solange bestehen, bis durch einen neuen Anwahl-Befehl ein anderes E/A-Gerät angewählt wird.

▷ Grundform: a 030. ($a \geq 4000$)

Wichtige Übertragungskanäle und deren Anwahl siehe Anlage 5.
Die Unterscheidung zwischen Eingabe und Ausgabe erfolgt während der Befehlsabarbeitung (Ein- oder Ausgabebefehl).

(22) Beispiel: Die zu errechnenden Werte sollen in einer Tabelle dargestellt werden, die über das Schreibwerk 2 ausgegeben wird.

32		
33	00, ,4400,030	AW K3
34		

3.7.2. Die Ausgabe eines Zeichens

Der Ausgabe-Halt-Befehl wird auch verwendet, um einzelne alpha-numerische oder Sonderzeichen auszugeben. Hierbei muß für jedes Zeichen ein Befehl programmiert werden. Der Operandenteil des Befehls beinhaltet dabei die Verschlüsselung (Ausgabeadresse) des auszugebenden Zeichens laut dem verwendeten Code. Für die Ausgabe über Schreibwerk und LB-Stanzer wird der R 300-Code verwendet (siehe Anlage 5).

▷ Grundform: a 030. (a - Ausgabeadresse des auszugebenden Zeichens laut verwendeten Code)

(23) Beispiel: Als Tabellenkopf der zu druckenden Tabelle sollen folgende Zeichen ausgegeben werden:
WRZL - TAB - ZWR - ZWR - x - 1 - .....

032			
033	00.	.4400,030	AW K3 (SW2)
034	00.	.2720,030	WRZL
035	00.	.1170,030	KB
036	00.	.1550,030	TAB
037	00.	. 320,030	ZWR
040	00.	. 320,030	ZWR
041	00.	.1470,030	x
042	00.	. 10,030	1
043			

3.8. Korrekturen im Programm

3.8.1. Leer-Befehle

Wenn beim Testen des Programms festgestellt wird, daß im Programm Befehle enthalten sind, die zuviel sind, gibt es die Möglichkeit, an dieser Stelle Leer-Befehle einzusetzen. Leer-Befehle löschen den Ac, auf das Speicherfeld, das als Operand angegeben wurde , haben diese Befehle keinen Einfluß.
Durch das Einschieben der Leer-Befehle ist es nicht notwendig, das gesamte Programm zu ändern. Das ist vor allem vorteilhaft, wenn viele Verzweigungen im Programm enthalten sind.

▷ Grundform: a 201.
a 403.
a 505.
a 601.
a 705.

Diese Befehle haben alle die gleiche Wirkung, es gibt außer den aufgezählten Befehlen weitere Leer-Befehle.

(24) Beispiel: Auf den Speicherplätzen 517 und 520 steht 2 mal der Befehl 00 320 030. (Ausgabe eines Leerzeichens) Beim Test wird festgestellt, daß diese 2 Leerzeichen nicht in das Druckbild passen.
Das ursprüngliche Programm hatte folgendes Aussehen:

515			
516	00	320 030	ZWR
517	00	320 030	ZWR
520	00	320 030	ZWR
521	00	570 030	GB
522			

Nach der Korrektur sieht das Programm folgendermaßen aus:

515	, , ,	
516	00, , 320,030	ZWR
517	01, , 25,201	Leer-Befehl
520	01, , 25,403	Leer-Befehl
521	00, , 570,030	GB
522	, , ,	

3.8.2. Das Überspringen von Befehlen

Eine weitere Möglichkeit Befehle unwirksam zu machen, ohne das gesamte Programm zu ändern, ist das Überspringen mit dem unbedingten Sprungbefehl.

(25) Beispiel: Problem und ursprüngliches Programm siehe Beispiel 24.
Das korrigierte Programm hat folgendes Aussehen:

515	, , ⋮ ,	
516	00, ,320 ,030	
517	01, ,521 ,000	Überspringen-520-
520	00, ,320 ,030	
521	00, ,570 ,030	
522	, , ⋮ ,	

3.8.3. Das Einfügen von Befehlen

Um ein Neuerarbeiten des Programms zu umgehen, springt man an der Stelle, an der noch ein oder mehrere Befehle eingefügt werden sollen, ans Programmende, programmiert dort die fehlenden Befehle und springt dann zu dem dem Sprungbefehl folgenden Befehl zurück.

(26) Beispiel: Bevor mit der Ausgabe des Tabellenkopfes begonnen wird, soll dem Bediener mittels eines bedingten Stop-Befehls die Möglichkeit gegeben werden, ein neues Blatt in das Schreibwerk einzuspannen. Es ist also nachträglich ein bedingter Stop einzufügen.
Ursprüngliches Programm:

32	, , ⋮ ,	
33	00, ,4400,030	AWK3
34	00, ,2720,030	WRZL
35	00, ,1170,030	KB
36	, , ⋮ ,	

Korrigiertes Programm:

32	,	, : ,	
33	01,	, 551, 000	
34	00,	,2720, 030	WRZL
35	00,	,1170, 030	KB
.	,	, : ,	
.	,	, : ,	
550	01,	, 020	Stop.
551	00,	,4400, 030	AW K3
552	01,	, 060	BS
553	01,	, 34, 000	

4. Die Unterprogrammtechnik

Bibliotheks- und Unterprogramme sind Programme, die bei immer wiederkehrenden Rechenvorgängen oder Verfahren Anwendung finden, d.h. sie werden angelegt, um sich häufig wiederholende Programmschritte einzusparen. Unterprogramme (UP) sind so programmiert, daß sie an beliebiger Stelle im Hauptprogramm (HP) eingesetzt bzw. an beliebiger Stelle auf der Interntrommel abgespeichert und dort vom Hauptprogramm "angesprungen" werden. Deshalb werden sie relativ programmiert.

Um ein UP anspringen zu können, müssen festgelegte Anschlußbedingungen erfüllt werden:

- Versorgungszellen müssen gefüllt sein (Bereitstellung der Daten);
- Ansprung des UP;
- Übergabewerte dem HP zurückübermitteln (Ergebnisse);
- Rücksprung ins HP.

Für allgemeine Unterprogramme gilt:

- Zur Bereitstellung der Parameter werden im allgemeinen die Register 1 - 3 genutzt.
- Wenn das Ergebnis nur aus einer Zahl besteht, wird diese dem HP im Ac übermittelt, anderenfalls kann ebenfalls auf die Register 1 - 3 zurückgegriffen werden.

Für elementare Rechenprogramme gilt:

- Beide Operanden sind in festgelegten Speicherzellen, den sogenannten Registern R1 - R3, bereitzustellen.
- Das Ergebnis wird im allgemeinen im Ac übermittelt.

Für Ein-/Ausgabe-Unterprogramme gilt:

- Die Anwahl des Ein-/ bzw. Ausgabe-Gerätes (E/A-Gerät) erfolgt im HP.

- Bei Ausgabe-Unterprogrammen ist der auszugebende Wert im R1 bereitzustellen.

Unterprogramme werden mit einem "Sprung mit Rückkehrabsicht" angesprungen.

▷ Grundform: a 025.

Dieser Befehl bewirkt, daß in den Ac die Adresse des Befehls "Sprung mit Rückkehrabsicht" um eins erhöht als unbedingter Sprungbefehl transportiert wird (<Ac> : = >B< + 1 000.) und anschließend ein unbedingter Sprung zu der im Adreßteil des Befehls angegebenen Speicherzelle erfolgt.
Der erste Befehl im Unterprogramm muß die Sicherung des Rücksprungs sein, der letzte Befehl der Rücksprung selbst.
Auch innerhalb eines Unterprogramms können wieder Unterprogramme Verwendung finden, man spricht hierbei von Unter-Unterprogrammen (UUP).

Abb. 12 Register, die zum Bereitstellen der Parameter für Unterprogramme verwendet werden

Register 1 - Speicherplatz 7773
Register 2 - Speicherplatz 7776
Register 3 - Speicherplatz 7774

4.1. Das Anlegen von Unterprogrammen

Unterprogramme können Teil des HP sein, sie können aber auch separat programmiert und abgespeichert werden. Die benötigten Parameter werden in der Regel in den Registern bereitgestellt (der Ac wird zur Sicherung des Rücksprungs benötigt), die Ergebnisse werden im Ac oder in den Registern dem Hauptprogramm übergeben. Wenn mit separaten Unterprogrammen gearbeitet wird, ist zu beachten, daß innerhalb eines Hauptprogramms nur 8 unterschiedliche Unterprogramme angesprungen werden können (8 verfügbare freie Leitadressen).

(27) Beispiel: Von den Geschwindigkeiten v_x und v_y soll der Betrag gebildet werden, dabei soll nach der Formel

$$v_{x,y} = \sqrt{v^2_{x,y}}$$

vorgegangen werden.
Das Unterprogramm ist als Teil des Hauptprogramms zu programmieren, die Bereitstellung des Parameters (v_x bzw. v_y) erfolgt über das Register 1 (Speicherplatz 7773), das Ergebnis soll im Ac an das Hauptprogramm übergeben werden.

6		
357	01, 13, 211	<Ac> := Vx
360	00, 7773, 240	<R1> := Vx
361	01, 370, 025	Vx := |Vx|
362	01, 13, 240	Vx → >13<
363	01, 14, 211	<Ac> := Vy
364	00, 7773, 240	<R1> := Vy
365	01, 370, 025	Vx := |Vx|
366	01, 14, 240	Vy → >14<
367	01, 376, 000	
370	01, 375, 240	Sicherung Rücksp
371	00, 7773, 211	
375	00,	Rücksprung

Separate Unterprogramme, wie z.B. das Unterprogramm $\sqrt{x}$, werden folgendermaßen angesprungen:

(28) Beispiel:

373		
374	02, 025	<Ac> := $\sqrt{<R_1>}$
375		

Die Ansprungadresse ist in der Regieanweisung für das jeweilige Unterprogramm festgelegt.
Jedes Unterprogramm muß dabei ein anderes Kennzeichen 02 - 09 erhalten, damit der Rechner einwandfrei identifizieren kann, welches UP er an dieser Stelle anspringen soll. Wenn z.B. im gleichen HP noch das UP sin x angesprungen werden soll, muß dieses mit dem Befehl

567		
570	03, 025	sin x
571		

geschehen. Für den Test des Programms ist es notwendig, daß dem Bediener eine Aufstellung der verwendeten UP und der zugeordneten Kennzeichen bereitgestellt wird.

Alle Service-Unterprogramme und Bibliotheksunterprogramme werden separat abgespeichert und müssen ein eigenes Kennzeichen zugeordnet bekommen.

4.2. Programmieren mit Interpretiersystem

Eine besondere Form der Unterprogramme sind die Interpretiersysteme. Für den C8205 Z stehen verschiedene Interpretiersysteme zur Verfügung, im folgenden wird das BIS (Bausteininterpretiersystem) erläutert, da es das gebräuchlichste Interpretiersystem in der Ausbildung ist.
Das BIS besteht aus leistungsstarken Bausteinen für die Gleitkomma- und Festkommarechnung. Dieses System kann man sich somit problembezogen selbst zusammenstellen. Das BIS wird auf sperrbaren Speicherplätzen abgespeichert, Arbeits- und Ansprungzellen sind zur Erleichterung der Programmierung fest definiert. Der Ansprung erfolgt ebenfalls mit dem Befehl

a 025.

Im folgenden eine Zusammenstellung der wichtigsten BIS-Bausteine.

4.2.1. Eingabe-Unterprogramme

- Eingabe einer Dezimalzahl (interne Konvertierung in eine Gleitkommazahl (GKZ)) - Worteingabe

OO 7574 025.

Nach der Eingabe des Wortes steht dieses im Ac und muß vom HP weiterverarbeitet werden. Die Eingabe jeder Dezimalzahl muß mit einem Punkt abgeschlossen werden.

- Eingabe von Dezimalzahlen im Satz (interne Konvertierung in GKZ)

OO 7566 025.

Vor Eingabe des Datensatzes muß die Versorgungszelle 7745 mit einem Abspeicherbefehl gefüllt werden, der festlegt, ab welcher Adresse a die Abspeicherung beginnen soll. Form des Abspeicherbefehls

a 240.

Jede Dezimalzahl ist mit einem Punkt abzuschließen, jeder Datensatz zusätzlich mit einer Satzmarke (WRZL) und jeder Datenblock mit einer Blockmarke (:).

Achtung! Der Rechner beendet die Eingabe erst, wenn er eine Satz- oder Blockmarke erkennt, beim Fehlen dieser Marken kann es zum Überschreiben der nachfolgenden Speicherzellen und damit zur Zerstörung des Programms kommen.

Wenn der Rechner eine Satzmarke (WRZL) erkennt, beendet er die Eingabe und springt zu dem Befehl, der unmittelbar dem Eingabe-Befehl folgt (>B< : = >B< + 1). Erkennt der Rechner eine Blockmarke,

beendet er ebenfalls die Eingabe und überspringt den Befehl, der dem Eingabe-Befehl folgt (>B< : = >B< + 2).
Diese Besonderheiten sind beim Programmieren von Satzeingaben unbedingt zu beachten, mit ihnen ist der ordnungsgemäße Abschluß des Programms möglich. Im Normalfall ist für den Datensatz, an dessen Ende eine Blockmarke steht, ebenfalls noch eine Berechnung durchzuführen. Damit der Rechner aber auch am Ende des Programms noch erkennt, daß er soeben den letzten Datensatz eines Blockes berechnet hat, ist das Anlegen eines Merkfeldes, des sogenannten Schalters, erforderlich. Dieser Schalter wird auf einem Speicherfeld angelegt, in der Regel wird es mit OO. definiert. Wenn der Rechner eine Satzmarke erkennt, bleibt der alte Inhalt des Schalters erhalten, sobald eine Blockmarke eingelesen wird, wird ein besonderes Zeichen, z.B. eine "Eins", auf dieses Speicherfeld transportiert.
Am Ende der Berechnungen wird dieser Schalter abgefragt - ist er "Null", so wird der nächste Datensatz eingelesen, steht ein anderes Zeichen auf diesem Speicherplatz, so wird das Programm beendet.

(29) Beispiel: Die Werte x_1, y_1 und z_1 sind im Satz einzulesen. Ihre Abspeicherung soll ab dem Speicherfeld 1 beginnen. Wenn eine Blockmarke erkannt wird, soll der Schalter von "null" auf "eins" umgestellt werden. Am Ende des Programms wird dieser Schalter abgefragt, steht er auf "eins", so wird das Programm beendet.

114	, , ,	
115	01, , 31,211	lesen Abspeicherbefehl
116	00, ,7745,240	Abspeicherbefehl >7745<
117	00, ,4500,030	AW KZ
120	00, ,7566,025	Eingabe P_1
121	01, , 124,000	bei Satzmarke
122	01, , 22,211	bei Blockmarke
123	01, , 20,240	1"→Schalter1
124	, , ,	

001 - x_1
002 - y_1
003 - z_1
020 - Schalter I ("0")
022 - "1"
023 - Abspeicherbefehl I
01 1 240.

4.2.2. Ausgabe-Unterprogramme

- Ausgabe einer GKZ als Dezimalzahl ohne Runden

00 p_2p_1 7560 025.

- Ausgabe einer GKZ als Dezimalzahl mit Runden

00 p_2p_1 7561 025.

Die auszugebende Zahl muß im Register 1 bereitgestellt werden.
Für die Ausgabe sind die Parameter p_1 und p_2 erforderlich, sie haben folgende Bedeutung:
p_1 - auszugebende Stellen nach dem Komma;
p_2 - auszugebende Stellen vor dem Komma.

$$0 \leqq p_{1;2} \leqq 15$$

Wenn die zu druckende Zahl mehr Stellen vor dem Komma besitzt, als p_2 angibt, so wird "zg" ("zu groß") ausgegeben.
Führende Nullen werden bei der Ausgabe unterdrückt, für sie werden Leerzeichen ausgegeben. Die Ausgabe beginnt mit der ersten von Null verschiedenen Zahl, spätestens aber mit der ersten Stelle vor dem Komma.
Die Parameter p_1 und p_2 werden folgendermaßen kodiert:

(30) Beispiel: Die errechneten Werte x_n sollen mit 6 Stellen vor und keiner Stelle nach dem Komma gerundet über das Schreibwerk 2 ausgegeben werden. Die Werte stehen auf dem Speicherplatz 4.

	p_2	p_1
geforderte Stellen (DZ)	6	0
geforderte Stellen als Dualzahl in Tetraden	0 1 1 0	0 0 0 0
Teilung in Triaden	0 1 | 1 0 0	0 0 0
Konvertierung in eine Oktalzahl	1 | 4	0
Parameter p_2p_1	140	

337	, , ,	
340	01, , 4,211	<Ac>:=<x_n>
341	00, ,7773,240	<R1>:=<x_n>
342	00,140,7561,025	A x_n
343	, , ,	

4.2.3. Gleitkomma-Rechenoperationen

Für die Rechenoperationen werden die Register zur Parameterversorgung genutzt.
Bei Operationen ohne Rückspeicherung steht das Ergebnis nur im Ac, bei Operationen mit Rückspeichern steht das Ergebnis außerdem noch im Register 1.
Bei Zahlenbereichsüberschreitung (siehe Punkt 2.1.2. - Gleitkommazahlen) tritt ein Überlauf auf (Anzeige FÜ an der Konsole).

Ansprungbefehl	Gleitkommaoperation
00 7553 025	<R1> : = <Ac> : = <R1> + <R2>
00 7552 025	<Ac> : = <R1> + <R2>
00 7551 025	<R1> : = <Ac> : = <R2> - <R1>
00 7550 025	<Ac> : = <R2> - <R1>
00 7547 025	<R1> : = <Ac> : = <R1> x <R3>
00 7546 025	<Ac> : = <R1> x <R3>
00 7545 025	<R1> : = <Ac> : = <R1> : <R3>
00 7544 025	<Ac> : = <R1> : <R3>

Abb. 13 Übersicht über GK-Rechenoperationen

Bei der Subtraktion und Division ist auf das richtige Besetzen der Register zu achten, ansonsten kommt es zu falschen Ergebnissen.

(31) Beispiel: Es soll der Wert Δx errechnet werden. Er wird nach der Formel

$$\Delta x := x_2 - x_1$$

gebildet und wird nach der Berechnung auf dem Speicherfeld 7 abgespeichert. Die Werte für x_1 und x_2 sind eingelesen. Die Addition soll mit Rückspeichern erfolgen, da der Wert Δx für die nachfolgende Rechenoperation im R1 benötigt wird.

204	⋮	
205	01 , 1 , 211	<Ac> := x_1
206	00 , 7773 , 240	<R_1> := x_1
207	01 , 4 , 211	<Ac> := x_2
210	00 , 7776 , 240	<R_1> := x_2
211	00 , 7551 , 025	<Δx> := <x_2> - <x_1>
212	01 , 7 , 240	Δx → >7<
	⋮	

001 - x_1
004 - x_2
007 - Δx

Die anderen Rechenoperationen können analog diesem Beispiel programmiert werden, es ist aber auf die für die Operation geforderten Register (R2 oder R3) zu achten.

Anlage 1

Programm
für Cellatron C 8205 Z

Problem: Extrapolation des Zielweges

Forderungen: 1. BIS-GK- Komplex

2. UP $\sqrt{x}$ (Kz. 02)

System/Sprache: BIS

Anlage:
Programmablaufplan 3 Blatt
Programmformular 12 Blatt

Bearbeiter: Rüdiger
Datum: 28.12.1982
Reg.-Nr.:

1. Aufgabenstellung

Im AFS wird eine Zielwegextrapolation zur Erleichterung der Entschlußfassung durch den Kommandeur durchgeführt. Diese Berechnung soll mit einem Programm für den C8205 Z simuliert werden.
Das Programm soll an Hand der eingelesenen kartesischen Koordinaten $P_1\ (x_1, y_1, z_1)$ und $P_2\ (x_2, y_2, z_2)$ die Koordinaten der 10 nächsten Standpunkte $P_n\ (x_n, y_n, z_n)$ berechnen und ausgeben. Das Zeitintervall soll eine Minute betragen.
Nach jeder neuen Positionsmeldung des Flugzeuges soll die Extrapolation wiederholt werden.
Am Ende der Berechnung der 10 Werte soll der Rechner die Geschwindigkeiten v_G, v_x, v_y und v_z ausgeben.

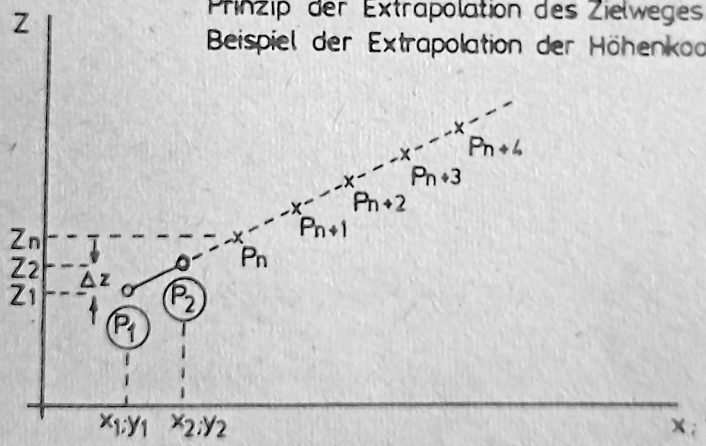
2. Problemanalyse

geg. $P_1\ (x_1, y_1, z_1)$
$P_2\ (x_2, y_2, z_2)$

ges. $P_n\ (x_n, y_n, z_n)$
v, v_x, v_y, v_z

Skizze:

Prinzip der Extrapolation des Zielweges am Beispiel der Extrapolation der Höhenkoordinate



Besondere Bedingungen, die durch das Programm beachtet werden müssen:

1. Die extrapolierte Höhenkoordinate kann kleiner als Null werden. In diesem Fall ist das Programm abzubrechen und es soll eine Fehlerausschrift ausgegeben werden.
Anschließend soll der Bediener gefragt werden, ob das Programm mit der Eingabe eines neuen Zieles fortgesetzt werden soll.

2. Durch das Programm soll es möglich sein, entweder für einen neuen realen Standort des Flugzeuges oder für ein anderes Flugzeug die Extrapolation durchzuführen.
3. Die Maßeinheit der einzulesenden Koordinaten soll Meter sein.
4. Die berechneten Geschwindigkeiten sollen in km x h^{-1} ausgegeben werden, von v, v_x und v_y ist der Betrag zu bilden.
5. Die Koordinaten x und y können negativ sein.

3. Modell

Voraussetzung: Das Zeitintervall zwischen 2 eingelesenen Koordinaten beträgt eine Minute.

- Berechnung P_n:

$$\Delta x = x_2 - x_1$$

$$x_n = x_2 + \Delta x$$

$$\Delta y = y_2 - y_2$$

$$y_n = y_2 - \Delta y$$

$$\Delta z = z_2 - z_1$$

$$z_n = z_2 + \Delta z$$

- Berechnung v, v_x, v_y, v_z

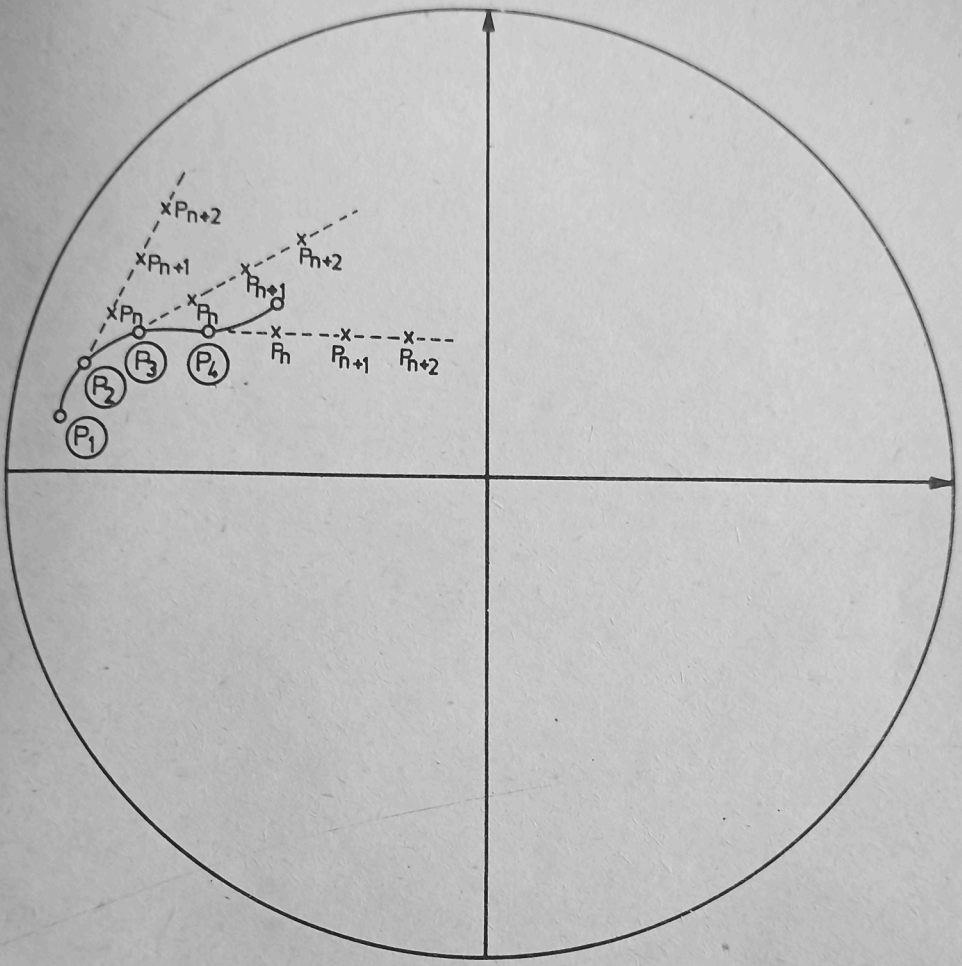
$$v_x \ (km \times h^{-1}) = \frac{\Delta x \ (m \times min^{-1})}{1000} \times 60 = \Delta x \times 0{,}06$$

$$v_y \ (km \times h^{-1}) = \frac{\Delta y \ (m \times min^{-1})}{1000} \times 60 = \Delta y \times 0{,}06$$

$$v_z \ (km \times h^{-1}) = \frac{\Delta z \ (m \times min^{-1})}{1000} \times 60 = \Delta z \times 0{,}06$$

$$v \ (km \times h^{-1}) = \frac{\sqrt{(\Delta x)^2 + (\Delta y)^2} \ (m \times min^{-1})}{1000} \times 60$$

$$= \sqrt{(\Delta x)^2 + (\Delta y)^2} \times 0{,}06$$



$P_1 - P_2 - P_3 - P_4 \ldots$ realer Zielweg

$P_n - P_{n+1} - P_{n+2} - \ldots$ extrapolierter Zielweg

4. Algorithmisierung

PAP siehe Seite 42.

Trockentest

geg. $x_1 = 100000$ m
$y_1 = 200000$ m
$z_1 = 10000$ m

$x_2 = 110000$ m
$y_2 = 210000$ m
$z_2 = 9500$ m

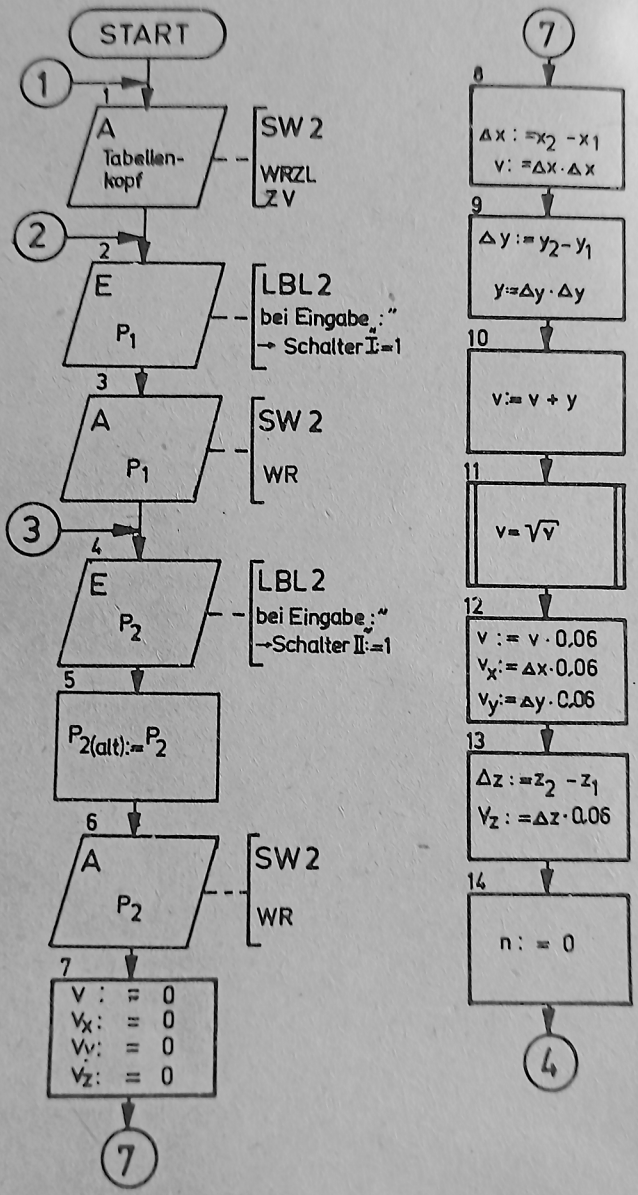
ges. P_n (n = 1 - 10)
v, v_x, v_y, v_z

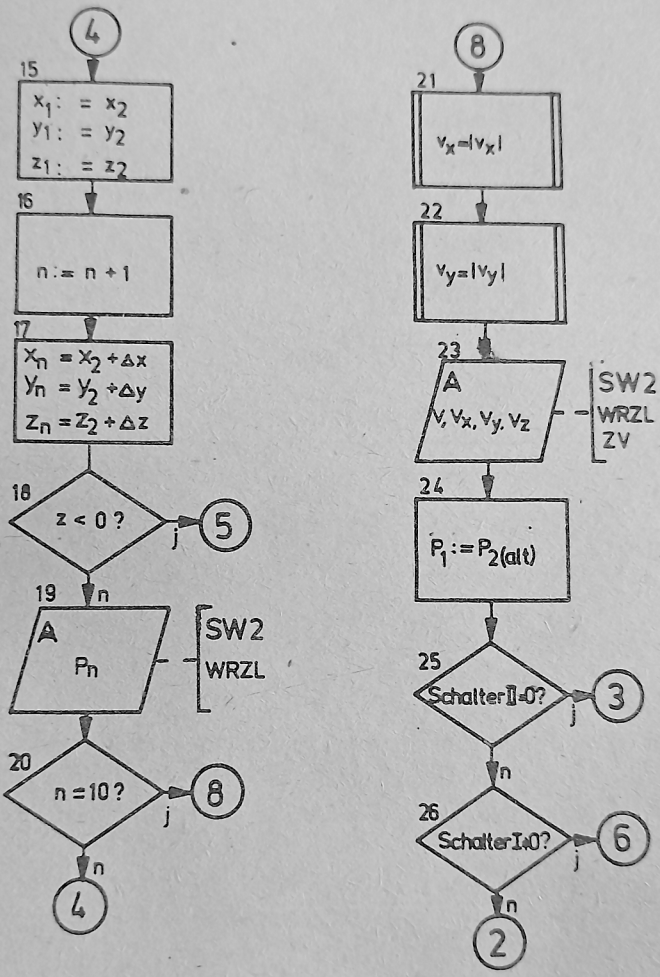
Lösung nach PAP

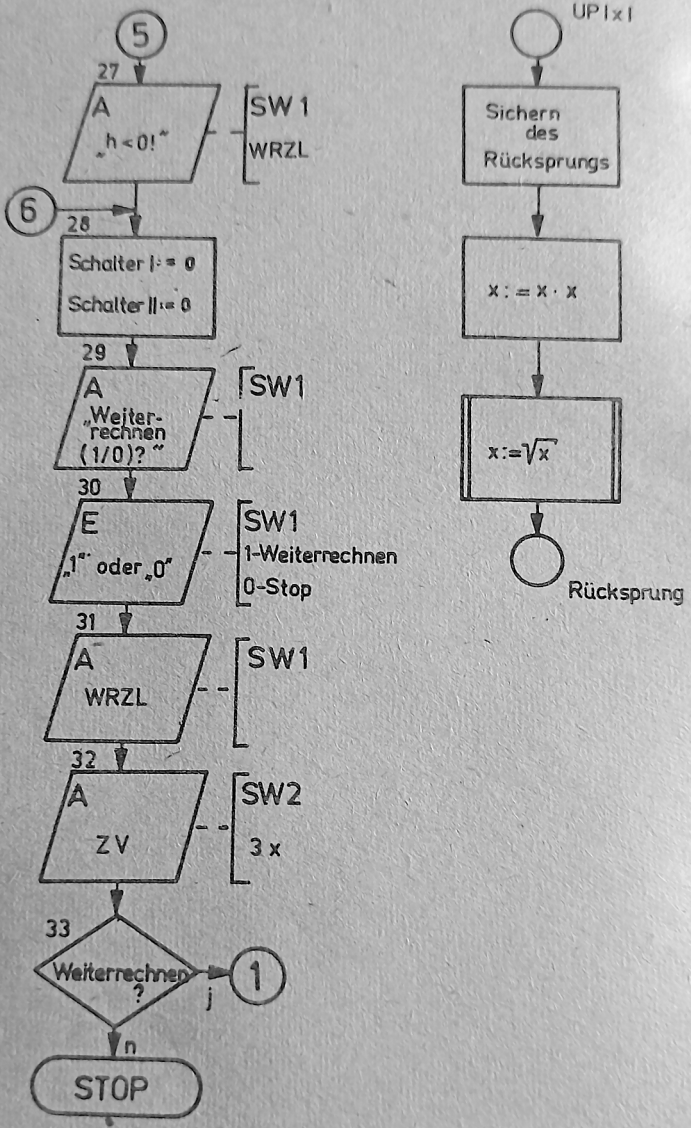
Schritt-Nr.	Operation
8	$\Delta x = 110000$ m - 100000 m
	$= 10000$ m
	$v = 10000$ m x 10000 m
	$= 100000000$ m^2
9	$\Delta y = 210000$ m - 200000 m
	$= 10000$ m
	$y = 10000$ m x 10000 m
	$= 100000000$ m^2
1	$v = 100000000$ m^2 + 100000000 m^2
	$= 200000000$ m^2
11	$v = 200000000$ m^2
	14142 m
12	$v = 14142$ m x 0,06
	849 km x h^{-1}
	$v_x = 10000$ m x 0,06
	$= 600$ km x h^{-1}
	$v_y = 10000$ m x 0,06
	$= 600$ km x h^{-1}
13	$\Delta z = 9500$ m - 10000 m
	$= -500$ m
	$v_z = -500$ m x 0,06
	$= -30$ km x h^{-1}

Programmablaufplan

Extrapolation des Zielweges




4
15
$x_1 := x_2$
$y_1 := y_2$
$z_1 := z_2$
16
n := n + 1
17
$x_n = x_2 + \Delta x$
$y_n = y_2 + \Delta y$
$z_n = z_2 + \Delta z$
18
z < 0 ?
j
5
19
n
A
P_n
SW2
WRZL
20
n = 10 ?
j
8
n
4
8
21
$v_x = |v_x|$
22
$v_y = |v_y|$
23
A
V, V_x, V_y, V_z
SW2
WRZL
ZV
24
$P_1 := P_{2(alt)}$
25
Schalter II = 0?
j
3
n
26
Schalter I = 0?
j
6
n
2


5
27
A
„h<0!"
SW1
WRZL
6
28
Schalter I := 0
Schalter II := 0
29
A
„Weiter-rechnen (1/0)?"
SW1
30
E
„1" oder „0"
SW1
1-Weiterrechnen
0-Stop
31
A
WRZL
SW1
32
A
ZV
SW2
3x
33
Weiterrechnen ?
j
1
n
STOP
UP Ix I
Sichern des Rücksprungs
x := x · x
x := √x
Rücksprung

Schritt-Nr.	Operation
16	$n = 0 + 1$ $= 1$
17	$x_n = 110000\ m + 10000\ m$ $= 120000\ m$ $y_n = 210000\ m + 10000\ m$ $= 220000\ m$ $z_n = 9500\ m + (-500\ m)$ $= 9000\ m$ Berechnung der restlichen 9 P_n
21	$v_x = 600\ km \times h^{-1} = 600\ km \times h^{-1}$
22	$v_y = 600\ km \times h^{-1} = 600\ km \times h^{-1}$

5. Programmierung

Programm siehe Seite 46 ff

Bedingungen für die Abarbeitung des Programms:

1. Ausgabe der Tabelle über das Schreibwerk 2.
2. Dialog mit dem Bediener über das SW 1.
3. Eingabe der Koordinaten über LBL 2.
4. Die Daten müssen in folgender Reihenfolge abgelocht werden:

$x_1 \cdot y_1 \cdot z_1 \cdot$WRZL $x_2 \cdot y_2 \cdot z_2 \cdot$WRZL $x_2 \cdot y_2 \cdot z_2 \cdot$WRZL....

Wenn die Koordinaten P_1 für ein neues Flugzeug eingelesen werden sollen, muß der vorherige Datensatz mit einer Blockmarke statt dem WRZL beendet werden.
Werden die Daten für das letzte Flugzeug eingelesen, wird die Koordinate P_1 auch mit einer Blockmarke an Stelle der Satzmarke abgeschlossen.

5. In das SW 2 ist ein DIN A3 Blatt im Querformat einzuspannen.
6. Das Programm arbeitet mit 9 Tabulatoren, diese sollten aller 15 Schreibschritte gesetzt werden. Der Abstand zwischen dem 3. und 4. sowie dem 6. und 7. Tabulator sollte 20 Schreibschritte betragen.
7. Wenn der Schalter BS in der unteren Stellung eingerastet ist, ist es möglich, vor der Ausgabe eines neuen Tabellenkopfes gegebenfalls ein neues Blatt einzuspannen.
8. Das Programm wird beendet, wenn auf die Frage "Weiterrechnen (1/0)?" mit "0" geantwortet wird, ansonsten wird das Programm mit der Ausgabe des Tabellenkopfes fortgesetzt.

Testbeispiel siehe Seite 58/59

C 8205	Programm: Extrapolation des Zielweges	Nr.
Befehlsfolge		

Bearbeiter: Rüdiger | Datum: 28.12.1982 | Blatt 1 von 12

Sp.	Nr.	Kz.	<AC>	Bemerkungen <R1> <R2> <R3> Schritt-Nr.
	00 0	01. . 33.000		
	1	00. . . x_1		
	2	00. . . y_2		
	3	00. . . z_1		
	4	00. . . x_2		
	5	00. . . y_2		
	6	00. . . z_2		
	7	00. . . Δ x		
	01 0	00. . . Δ y		
	1	00. . . Δ z		
	2	00. . . v		
	3	00. . . v_x		
	4	00. . . v_y		
	5	00. . . v_z		
	6	00. . . y		
	7	00. . . n		
	02 0	00. . . S I		
	1	00. . . S II		
	2	31. . . "1"		
	3	31.0 . . "10"		
	4	30,0,06 . "0,6"		
	5	00. . . "0"		
	6	00. . . x_2 (alt)		
	7	00. . . y_2 (alt)		
	03 0	00. . . z_2 (alt)		
	1	01. . 1.240	Absp.Bef.1	
	2	01. . 4.240	Absp.Bef.2	
	3	01. . 551.000		
	4	00. . 2720.030	WRZL	
	5	00. . 1170.030	KB	
	6	00. . 1550.030	TAB	
	7	00. . 320.030	ZWR	

(1)

C 8205

Befehlsfolge

Programm

Nr.

Bearbeiter:

Datum:

Blatt 2 von 12

Sp.	Nr.	Kz.	<AC>	Bemerkungen <R1> <R2> <R3> Schritt-Z
	040	00. 320.030	ZWR	1
	1	00. 1470.030	x	
	2	00. 10.030	1	
	3	00. 1550.030	TAB	
	4	00. 320.030	ZWR	
	5	00. 320.030	ZWR	
	6	00. 1500.030	y	
	7	00. 10.030	1	
	050	00. 1550.030	TAB	
	1	00. 320.030	ZWR	
	2	00. 320.030	ZWR	
	3	00. 1710.030	z	
	4	00. 10.030	1	
	5	00. 1550.030	TAB	
	6	00. 320.030	ZWR	
	7	00. 320.030	ZWR	
	060	00. 1470.030	x	
	1	00. 20.030	2	
	2	00. 1550.030	TAB	
	3	00. 320.030	ZWR	
	4	00. 320.030	ZWR	
	5	00. 1500.030	y	
	6	00. 20.030	2	
	7	00. 1550.030	TAB	
	070	00. 320.030	ZWR	
	1	00. 320.030	ZWR	
	2	00. 1710.030	z	
	3	00. 20.030	2	
	4	00. 1550.030	TAB	
	5	00. 320.030	ZWR	
	6	00. 320.030	ZWR	
	7	00. 1470.030	x	

C 8205 — Befehlsfolge | Programm | Nr.

Bearbeiter: | Datum: | Blatt 3 von 12

Sp.	Nr.	Kz.	<AC>	Bemerkungen <R1> <R2> <R3>	Schritt-Nr.
	10 0	00. .1050.030	n		1
	1	00. .1550.030	TAB		
	2	00. .320.030	ZWR		
	3	00. .320.030	ZWR		
	4	00. .1500.030	y		
	5	00. .1050.030	n		
	6	00. .1550.030	TAB		
	7	00. .320.030	ZWR		
	11 0	00. .320.030	ZWR		
	1	00. .1710.030	z		
	2	00. .1050.030	n		
	3	00. .2720.030	WRZL		
2 →	4	00. .1720.030	ZV		
	5	01. .31.211	Absp.Bef.1		2
	6	00. .7745.240			
	7	00. .4500.030	AW K2		
	12 0	00. .7566.025	E P_1		
	1	01. .124.000			
	2	01. .22.211	S 1 = "1"		
	3	01. .20.240			
→	4	00. .4400.030	AW K3		3
	5	00. .1550.030	TAB		
	6	01. .1.211	x_1		
	7	00. .7773.240		x_1	
	13 0	00.140 .7560.025	A x_1		
	1	00. .1550.030			
	2	01. .-2.211	y_1		
	3	00. .7773.240		y_1	
	4	00. .7560.025	A y_1		
	5	00. .1550.030	TAB		
	6	01. .3.211	z_1		
	7	00. .7773.240		z_1	

C 8205 Befehlsfolge	Programm	Nr.
Bearbeiter:	Datum:	Blatt 4 von 12

Sp.	Nr.	Kz.	<AC>	<R1>	<R2>	<R3>	Schritt-Nr.
	14 0	00.140 . 7560.025	A z_1				
	1	00. . 1120.030	WZ				
3	2	00. . 4500.030	AW K2				4
	3	01. . 32.211	Absp.Bef.2				
	4	00. . 7745.240					
	5	00. . 7566.025	E P_2				
	6	01. . 151.000					
	7	01. . 22.211	S II = "1"				
	15 0	01. . 21.240					
	1	01. . 4.211	x_2				5.1.
	2	01. . 26.240					
	3	01. . 5.211	y_2				5.2.
	4	01. . 27.240					
	5	01. . 6.211	z_2				5.3.
	6	01. . 30.240					
	7	00. . 4400.030	AW K3				6
	16 0	00. . 1550.030	TAB				
	1	00. . 1550.030	TAB				
	2	00. . 1550.030	TAB				
	3	00. . 1550.030	TAB				
	4	01. . 4.211	x_2				
	5	00. . 7773.240		x_2			
	6	00.140 . 7560.025	A x_2				
	7	00. . 1550.030	TAB				
	17 0	01. . 5.211	y_2				
	1	00. . 7773.240		y_2			
	2	00.140 . 7560.025	A y_2				
	3	00. . 1550.030	TAB				
	4	01. . 6.211	z_2				
	5	00. . 7773.240		z_2			
	6	00.140 . 7560.025	A z_2				
	7	00. . 1120.030	WR				

C 8205 Befehlsfolge	Programm	Nr.
Bearbeiter:	Datum:	Blatt 5 von 12

Sp.	Nr.	Kz.	<AC>	Bemerkungen <R1> <R2> <R3>	Schritt-Nr.
	20 0	01 · · 25·211	"0"		7.1.
	1	01 · · 12·240	v = 0		
	2	01 · · 13·240	$v_x = 0$		7.2.
	3	01 · · 14·241	$v_y = 0$		7.3.
	4	01 · · 15·241	$v_z = 0$		7.4.
	5	01 · · 1·211	x_1		8.1.
	6	00 · ·7773·240		x_1	
	7	01 · · 4.211	x_2		
	21 0	00 · ·7776·240		x_2	
	1	00 · ·7551·025	$\Delta x = x_2 - x_1$	Δx	
	2	01 · · 7·240			8.2.
	3	00 · ·7774·240		Δx	
	4	00 · ·7546·025	$v_G = (\Delta x)^2$		
	5	01 · · 12·240			
	6	01 · · 2·211	y_1		9.1.
	7	00 · ·7773·240		y_1	
	22 0	01 · · 5·211	y_2		
	1	00 · ·7776·240		y_2	
	2	00 · ·7551·025	$\Delta y = y_2 - y_1$	Δy	
	3	01 · · 10·240			
	4	00 · ·7774·240		Δy	9.2.
	5	00 · ·7547·025	$y = (\Delta y)^2$	y	
	6	01 · · 16·240			
	7	01 · · 12·211	v_G		10
	23 0	00 · ·7776·240		v	
	1	00 · ·7553·025	v = v + y		
	2	01 · · 12·240			
	3	02 · · ·025	$v = \sqrt{v}$		11
	4	01 · · 12·240			
	5	00 · ·7773·240		v	12.1.
	6	01 · · 24·211	0,06		
	7	00 · ·7774·240		0,06	

C 8205 | Programm | Nr.

Befehlsfolge

Bearbeiter: | Datum: | Blatt 6 von 12

Sp	Nr.	Kz.	<AC>	Bemerkungen <R1> <R2> <R3>	Schritt-Nr.
	24 0	00. .7546.025	v = v+0,06		
	1	01. 12.240			
	2	01. 7.211	Δx		12.2.
	3	00. .7773.240		Δx	
	4	01. 24.211	0,06		
	5	00. .7774.240		0,06	
	6	00. .7546.025	$v_x = \Delta x \cdot 0,06$		
	7	01. 13.240			
	25 0	01. 10.211	Δy		12.3.
	1	00. .7773.240		Δy	
	2	01. 24.211	0,06		
	3	00. .7774.240		0,06	
	4	00. .7546.025	$v_y = \Delta y \cdot 0,06$		
	5	01. 14.240			
	6	01. 3.211	z_1		13.1.
	7	00. .7773.240		z_1	
	26 0	01. 6.211	z_2		
	1	00. .7776.240		z_2	
	2	00. .7551.025	$\Delta z = z_2 - z_1$	Δz	
	3	01. 11.240			
	4	01. 24.211	0,06		13.2.
	5	00. .7774.240		0,06	
	6	00. .7546.025	$v_z = \Delta z \cdot 0,06$		
	7	01. 15.240			
4	27 0	01. 17.241	n = 0		14
	1	01. 4.211	x_2		
	2	01. 1.240	$x_1 = x_2$		15.1.
	3	01. 5.211	y_2		
	4	01. 2.240	$y_1 = y_2$		15.2.
	5	01. 6.211	z_2		15.3.
	6	01. 3.240	$z_1 = z_2$		
	7	01. 17.211	n		16

C 8205 Befehlsfolge	Programm	Nr.
Bearbeiter:	Datum:	Blatt 7 von 12

Sp.	Nr.	Kz.	<AC>	Bemerkungen <R1> <R2> <R3>
	30 0	00. .7773. 240		n
	1	01. . 22. 211	"1"	
	2	00. .7776. 240		"1"
	3	00. .7552. 025	n := n + 1	
	4	01. . 17. 240		
	5	01. . 1. 211	x_1	
	6	00. .7773. 240		x_1
	7	01. . 7. 211	Δx	
	31 0	00. .7776. 240		Δx
	1	00. .7552. 025	$x_2 := x_1 + \Delta x$	
	2	01. . 4. 240		
	3	01. . 2. 211	y_1	
	4	00. .7773. 240		y_1
	5	01. . 10. 211	Δy	
	6	00. .7776. 240		Δy
	7	00. .7552. 025	$y_2 := y_1 + \Delta y$	
	32 0	01. . 5. 240		
	1	01. . 3. 211	z_1	
	2	00. .7773. 240		z_1
	3	01. . 11. 211	Δz	
	4	00. .7776. 240		Δz
	5	00. .7552. 025	$z_2 := z_1 + \Delta z$	
	6	01. . 6. 240		
5	7	01. . 464. 040	$z < 0$?	
	33 0	00. .4400. 030	AW K3	
	1	00. .1550. 030	TAB	
	2	00. .1550. 030		
	3	00. .1550. 030		
	4	00. .1550. 030		
	5	00. .1550. 030		
	6	00. .1550. 030		
	7	00. .1550. 030		

Programm	Nr.

Datum: | Blatt 7 von 12

Kz.	<AC>	Bemerkungen <R1> <R2> <R3>	Schritt-Nr.
00. 7773. 240		n	16
01. 22. 211	"1"		
00. 7776. 240		"1"	
00. 7552. 025	n := n + 1		
01. 17. 240			
01. 1. 211	x_1		17.1.
00. 7773. 240		x_1	
01. 7. 211	Δx		
00. 7776. 240		Δx	
00. 7552. 025	$x_2 := x_1 + \Delta x$		
01. 4. 240			
01. 2. 211	y_1		17.2.
00. 7773. 240		y_1	
01. 10. 211	Δy		
00. 7776. 240		Δy	
00. 7552. 025	$y_2 := y_1 + \Delta y$		
01. 5. 240			
01. 3. 211	z_1		17.3.
00. 7773. 240		z_1	
01. 11. 211	Δz		
00. 7776. 240		Δz	
00. 7552. 025	$z_2 := z_1 + \Delta z$		
01. 6. 240			
01. 464. 040	$z < 0$?		18
00. 4400. 030	AW K3		19
00. 1550. 030	TAB		
00. 1550. 030			
00. 1550. 030			
00. 1550. 030			
00. 1550. 030			
00. 1550. 030			
00. 1550. 030			

C 8205 Befehlsfolge | Programm | Nr.

Bearbeiter: | Datum: | Blatt 8 von 12

Sp.	Nr.	Kz.	<AC>	Bemerkungen <R1> <R2> <R3>	Schritt-Nr.
	34 0	01. . 4. 211	x_2		19
	1	00. . 7773. 240		x_2	
	2	00. 140 . 7561. 025	A x_2		
	3	00. . 1550. 030	TAB		
	4	01. . 5. 211	y_2		
	5	00. . 7773. 240		y_2	
	6	00. 140 . 7561. 025	A y_2		
	7	00. . 1550. 030	TAB		
	35 0	01. . 6. 211	z_2		
	1	00. . 7773. 240		z_2	
	2	00. 140 . 7561. 025	A z_2		
	3	00. . 2720. 030	WRZL		
	4	01. . 17. 211	n		20
	5	01. . 23. 330	$n = n_{max}$?		
4	6	01. . 271. 160			
	7	01. . 13. 211	v_x		21
	36 0	00. . 7773. 240		v_x	
	1	01. . 370. 025	UP $\lvert v_x \rvert$		
	2	01. . 13. 240			
	3	01. . 14. 211	v_y		22
	4	00. . 7773. 240		v_y	
	5	01. . 370. 025	UP $\lvert v_y \rvert$		
	6	01. . 14. 240			
	7	01. . 376. 000			
	37 0	01. . 375. 240	UP $\lvert x \rvert$		
	1	00. . 7773. 211	x	x	
	2	00. . 7774. 240		x	
	3	00. . 7547. 025	$x = x^2$	x^2	
	4	02. . . 025	$x := \sqrt{x}$		
	5	00. . .	Rücksprung		
	6	00. . 4400. 030	AW K3		23
	7	00. . 1550. 030	TAB		

C 8205	Programm	Nr.
Befehlsfolge		
Bearbeiter:	Datum:	Blatt 9 von 12

Sp.	No.	Kz.	<AC>	Bemerkungen <R1> <R2> <R3> Sch
	40 0	00. .1550.030	TAB	
	1	00. .1550.030		
	2	00. .1550.030		
	3	00. .1550.030		
	4	00. .1650.030	V	
	5	00. . 320.030	ZWR	
	6	00. . 570.030	GB	
	7	00. . 400.030	=	
	41 0	00. .1170.030	KB	
	1	01. . 12.211	V	
	2	00. .7773.240		V
	3	00.120 .7561.025	A V	
	4	00. .1550.030	TAB	
	5	00. .1650.030	V	
	6	00. .1470.030	X	
	7	00. . 570.030	GB	
	42 0	00. . 400.030	=	
	1	00. .1170.030	KB	
	2	01. . 13.211	V_X	
	3	00. .7773.240		V_X
	4	00.120 .7561.025		
	5	00. .1550.030	TAB	
	6	00. .1650.030	V	
	7	00. . .1500.030	Y	
	43 0	00. . 570.030	GB	
	1	00. . 400.030	=	
	2	00. .1170.030	KB	
	3	01. . 14.211	V_Y	
	4	00. .7773.240		V_Y
	5	00.120 .7561.025	A V_Y	
	6	00. .1550.030	TAB	
	7	00. .1650.030	V	

C 8205	Programm	Nr.
Befehlsfolge		
Bearbeiter:	Datum:	Blatt 10 von 12

Sp.	Nr.	Kz.	<AC>	Bemerkungen <R1> <R2> <R3>	Schritt-Nr.
	44 0	00. . 1710. 030	Z		
	1	00. . 570. 030	GB		
	2	00. . 400. 030	=		
	3	00. . 1170. 030	KB		
	4	01. . 15 211	v_z		
	5	00. . 7773. 240		v_z	
	6	00 120 . 7561. 025	A v_z		
	7	00. . 2720. 030	WRZL		
	45 0	00. . 1720. 030	ZV		
	1	01. . 26 211	$P_1 := P_2$		24
	2	01. . 1 240			
	3	01. . 27 211			
	4	01. . 2 240			
	5	01. . 30 211			
	6	01. . 3 240			
	7	01. . 21 331	S II = 0 ?		25.1.
3	46 0	01. . 142 170			
	1	01. . 20 331	S I = 0 ?		26
2	2	01. . 114 170			
6	3	01. . 474 000			
	4	00. . 4500 030	AW K2		27
5	5	00. . 700 030	n		
	6	00. . 570 030	GB		
	7	00. . 10 030	<		
	47 0	00. . 1060 030	0		
	1	00. . 320 030	ZWR		
	2	00. . 730 030	!		
	3	00. . 2720 030	WRZL		
	4	01. . 20 241	S I := 0		28
	5	01. . 21 241	S II := 0		
	6	00. . 4500 030	AW K2		
	7	00. . 570 030	GB		

C 8205
Befehlsfolge

Programm

Nr.

Bearbeiter:

Datum:

Blatt 11 von 12

Sp.	Nr.	Ku.	<AC>	Bemerkungen <R1> <R2> <R3> Sprung u.
	50 0	00. 1660.030	w	29
	1	00. 1170.030	KB	
	2	00. 450.030	e	
	3	00. 510.030	i	
	4	00. 1630.030	t	
	5	00. 450.030	e	
	6	00. 1110.030	r	
	7	00. 1110.030	r	
	51 0	00. 450.030	e	
	1	00. 430.030	c	
	2	00. 700.030	h	
	3	00. 1050.030	n	
	4	00. 450.030	e	
	5	00. 1050.030	n	
	6	00. 320.030	ZWR	
	7	01. 25.201	Leer-Befehl	
	52 0	01. 25.403	Leer-Befehl	
	1	00. 570.030	GB	
	2	00. 150.030	(	
	3	00. 1170.030	KB	
	4	00. 10.030	1	
	5	00. 570.030	GB	
	6	00. 260.030	/	
	7	00. 1060.030	0	
	53 0	00. 1410.030	)	
	1	00. 320.030	ZWR	
	2	00. 1530.030	?	
	3	00. 1170.030	KB	
	4	00. 320.030	ZWR	
	5	00. 4700.030	AW KO	
	6	00. 7674.025	E 1 oder 0	
	7	01. 16.240		

C 8203 Befehlsfolge	Programm	Nr.
Bearbeiter:	Datum:	Blatt 12 von 12

Sp.	No.	Kz.	<AC>	Bemerkungen <R1> <R2> <R3>	Schritt-Nr.
	54 0	00. .4500.030	AW K2		31
	1	00. .2720.030	WRZL		
	2	00. .4400.030	AW K3		32
	3	00. .1720.030	ZV		
	4	00. .1720.030	ZV		
	5	00. .1720.030	ZV		
	6	01. . 16.331			33
(1)	7	01. . 33.160			
	55 0	01. . .020	Stop		
	1	00. .4400.030	AW K3		
	2	01. . .060	BS		
	3	01. . 34.000			
	4	. . .			
	5	. . .			
	6	. . .			
	7	. . .			
	0	. . .			
	1	. . .			
	2	. . .			
	3	. . .			
	4	. . .			
	5	. . .			
	6	. . .			
	7	. . .			
	0	. . .			
	1	. . .			
	2	. . .			
	3	. . .			
	4	. . .			
	5	. . .			
	6	. . .			
	7	. . .			
		. . .			
		. . .			
		. . .			
		. . .			
		. . .			

Cellatron C8205 Z

Berechnungsbeleg

x1	y1	a1	a2
100000	200000	10000	110000

v = 849 vx= 600

100000 220000 9250

v = 849

110000 200000

h =0!

Weiterrechnen (110) ? 0.

Berechnung durchgeführt _ _OS_Rüdiger_ _ _

Zur Demonstration der Funktion des Programms, wurden Tabelle und Service über ein Schreibwerk ausgegeben.

a2	xn	yn	[illegible]
9500	120000	220000	9000
	130000	230000	8500
	140000	240000	8000
	150000	250000	7500
	160000	260000	7000
	170000	270000	6500
	180000	280000	6000
	190000	290000	5500
	200000	300000	5000
	210000	310000	4500
vx= 600	vy= 600	vz= 30-	
8250	90000	230000	9000
	80000	240000	8750
	70000	250000	8500
	60000	260000	8250
	50000	270000	8000
	40000	280000	7750
	30000	290000	7500
	20000	300000	7250
	10000	310000	7000
	0	320000	6750
	vy= 600	vz= 15-	
y2			
210000	120000	180000	7250
	130000	160000	6250
	140000	140000	5250
	150000	120000	4250
	160000	100000	3250
	170000	80000	2250
	180000	60000	1250
	190000	40000	250

Datum: 15.03.83

Anlage 2: Zusammenstellung der wichtigsten Befehle

Befehl	Bedeutung
000	unbedingter Sprung >nächster Befehl< := a
020	unbedingter Stop
025	Sprung mit Rückkehrabsicht <Ac> := >B< + 1
030	Ausgabe-Halt-Befehl - Anwahl eines EA-Geräts - Ausgabe eines Zeichens
040	Negativsprung Sprung nach a, wenn <Ac> < 0
050	Positivsprung Sprung nach a, wenn <Ac> $\geq$ 0
060	bedingter Stop
140	Vergleichssprung 1 Sprung nach a, wenn <Ac> $\leq$ <a*>
150	Vergleichssprung 2 Sprung nach a, wenn <Ac> $\geq$ <a*>
160	Ungleichsprung Sprung nach a, wenn <Ac> $\neq$ <a*>
170	Vergleichssprung 3 Sprung nach a, wenn <Ac> = <a*>
201, 403, 505, 601, 705	Leer-Befehle <Ac> := 0
211	Lese-Befehl <Ac> := <a>
240	Transport-Befehl <a> := <Ac>
241	Lösch-Befehl <a> := <Ac> := 0
330	reines Vergleichersetzen <Ac> $\lessgtr$ <a*> ?
331	Vergleichersetzen <a*> $\lessgtr$ 0 ?

Anlage 3: Zusammenstellung der wichtigsten Unterprogramme

Eingabe - UP

7674 Eingabe einer DZ (int. Konv. in GKZ)

7666 Eingabe von DZ im Satz (int. Konv. in GKZ)

Ausgabe - UP

P_2P_1 7660 Ausgabe einer GKZ als DZ ohne Runden

P_2P_1 7661 Ausgabe einer GKZ als DZ mit Runden

GK - Rechenoperationen

7644			<Ac>	:=	<R1>	: <R3>
7645	<R1>	:=	<Ac>	:=	<R1>	: <R3>
7646			<Ac>	:=	<R1>	x <R3>
7647	<R1>	:=	<Ac>	:=	<R1>	x <R3>
7650			<Ac>	:=	<R2>	- <R1>
7651	<R1>	:=	<Ac>	:=	<R2>	- <R1>
7652			<Ac>	:=	<R1>	• <R2>
7653	<R1>	:=	<Ac>	:=	<R1>	• <R2>

Bibliotheksunterprogramme

Funktion	Eingabe	Ergebnis	Ansprung
e^x	R1	Ac	LA+0
ln x	R1	Ac	LA+0
sin x, cos x	(1) R1	Ac	LA+14 LA+0
tan x, cot x	(1) R1	Ac	LA+0 LA+35
arcsin x, arccos x	(1) R1	Ac	LA+0 LA+22
arctan x, arccot x	(1) R1	Ac	LA+0 LA+162
$\sqrt{x}$	R1	Ac	LA+0
$\sqrt[n]{x}$	x - LA+1 n - LA+2	Ac	LA+0
$\hat{\alpha} \to \alpha^\circ$ $\alpha^\circ \to \hat{\alpha}$	R1	Ac u. R1	LA+0 LA+7

(i) Der Winkel ist bei diesen Unterprogrammen in Bogenmaß anzugeben. Dazu ist der Aufruf des UP $\widehat{\alpha} \to \alpha^\circ / \alpha^\circ \to \widehat{\alpha}$ vor diesen UP notwendig.

Beispiel: Es ist der tan x zu errechnen.
Der Wert x steht auf dem Speicherplatz 1,
das UP $\alpha^\circ \to \widehat{\alpha}$ hat das Kennzeichen 02 und
das UP tan x hat das Kennzeichen 03.

7	, , : ,	
610	01 , , 1, 211	<A0> := x
611	00 , ,7773, 240	<R1> := x
612	02 , , 7, 025	$x^\circ \to \widehat{x}$
613	03 , , , 025	<A0> := tan x
4	, , : ,	

Anlage 4: Programmierungsvarianten für Vergleicher- und Sprungbefehle

Vergleicher-Sprungbefehl	a 330.		a 331.	
	Sprung nach a, wenn	nächster Befehl, wenn	Sprung nach a, wenn	nächster Befehl, wenn
a 140.	<Ac> ≦ <a*>	<Ac> > <a*>	<a*> ≦ 0	<a*> > 0
a 150.	<Ac> ≧ <a*>	<Ac> < <a*>	<a*> ≧ 0	<a*> < 0
a 160.	<Ac> ≠ <a*>	<Ac> = <a*>	<a*> ≠ 0	<a*> = 0
a 170.	<Ac> = <a*>	<Ac> ≠ <a*>	<a*> = 0	<a*> ≠ 0

Befehl	Sprung nach a, wenn	nächster Befehl, wenn
a 040.	<Ac> < 0	<Ac> ≧ 0
a 050.	<Ac> ≧ 0	<Ac> < 0

Anwendung siehe Seite 22 ff

Anlage 5: Ausgabeadressen des R-300-Code

Zeichen KB	Zeichen GB	Ausgabe-adresse
a	A	0610
b	B	0620
c	C	0430
d	D	0640
e	E	0450
f	F	0460
g	G	0670
h	H	0700
i	I	0510
j	J	1210
k	K	1220
l	L	1030
m	M	1240
n	N	1050
o	O	1060
p	P	1270
q	Q	1300
r	R	1110
s	S	1420
t	T	1630
u	U	1440
v	V	1650
w	W	1660
x	X	1470
y	Y	1500
z	Z	1710

Zeichen KB	Zeichen GB	Ausgabe-adresse
0	#	0200
1	<	0010
2	"	0020
3	≈	0230
4	[	0040
5	>	0250
6	/	0260
7	¬	0070
8	~	0100
9	]	0310
ZWR		0320
:	(	0150
+	=	0400
.	!	0730
-	△	1000
,	;	1600
/	)	1410
%	*	1740
'	?	1530
WRZL		2720
GB		0570
KB		1170
TAB		1550
IS		0160
IZ		3770
BM		3520
Transport		1770
IB		0760
WR		1120
ZV		1720
Transportloch im Vorspann		0000

ZWR - Zwischenraum
WRZL - Wagenrücklauf-Zeilenvorschub
WR - Wagenrücklauf
ZV - Zeilenvorschub
TAB - Tabulatorsprung
GB - Großbuchstaben

KB - Kleinbuchstaben
IZ - Irrung Zeichen
IS - Irrung Satz
IB - Irrung Block
BM - Blockmarke

Anwahlbefehle für die E/A-Geräte

Kanal	Eingabe	Ausgabe	Anwahlbefehl
0	Schreibwerk 1	LB-Stanzer 2	00 4700 030.
1	LB-Leser 1	LB-Stanzer 1	00 4600 030.
2	LB-Leser 2	Schreibwerk 1	00 4500 030.
3	frei	Schreibwerk 2	00 4400 030.
4	Universalausgabekanäle (LK-Leser und -Stanzer usw.)		00 4300 030.
5			00 4200 030.
6			00 4100 030.
7	Anschlußkanal DAR01600		00 4000 030.

Anlage 6: Literaturhinweise für tiefgreifendere Ausführungen über die Programmierung des C8205 Z

1. Programmierung und Bedienung der elektronischen Rechenanlage C8205 Z

   Heft 1 - 9
   VEB Rechenelektronik Meiningen/Zella-Mehlis

2. Reihe Automatisierungstechnik

   Hefte 5, 12, 52, 116 und 126

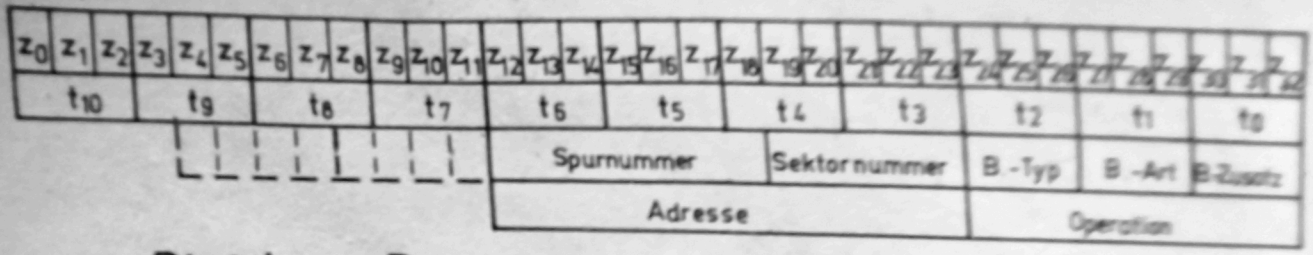
3. C8205 Bedienungsanleitung

4. C8205 Anleitung zur Programmierung in der Maschinensprache

5. C8205 Grundbegriffe der Programmierung

Literatur 3 - 5:

Ministerium für Nationale Verteidigung
Institut für Mechanisierung und Automatisierung der Truppenführung



Die interne Darstellung eines Befehls

Triade	t_{10}			t_9			t_8			t_7			t_6			t_5			t_4			t_3			t_2			t_1			t_0		
Bit	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9	z_{10}	z_{11}	z_{12}	z_{13}	z_{14}	z_{15}	z_{16}	z_{17}	z_{18}	z_{19}	z_{20}	z_{21}	z_{22}	z_{23}	z_{24}	z_{25}	z_{26}	z_{27}	z_{28}	z_{29}	z_{30}	z_{31}
Dualz.	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	1	0	1	0	1	0	1	0	0	0	1	0	0	0
Oktalz.	0			0			0			0			7			1			6			5			2			1			0		

Maschinenwort in Triodendarstellung (mit Beispiel)

Anlage 8: Abkürzungen und Symb

A	Operationsart oder Ausgabe
a	Adresse im Befehl (Operand)
a_r	Adresse im Befehl (Operand, mit dem ein Vergleich durchgeführt wird)
Ac	Akkumulator
AW	Anwahl
B	Befehlstyp
>B<	Adresse des Befehls
BIS	Bausteininterpretiersystem
BS	Bedingter Stop
DZ	Dezimalzahl
E	Eingabe
E/A-Gerät	Ein- oder Ausgabegerät
FKZ	Festkommazahl
GB	Großbuchstabe
GKZ	Gleitkommazahl
HLA	Hauptleitadresse
HP	Hauptprogramm
int.	intern
K0-K3	Kanal 0-3
KB	Kleinbuchstabe
Konv.	Konvertierung
Kz.	Kennzeichen
LA	Leitadresse
LB	Lochband
LBL	Lochbandleser
PAP	Programmablaufplan
R1-R3	Register 1-3
rel.	relativ
SW	Schreibwerk
TAB	Tabulatorsprung
t_{0-10}	Triade 0-10 des Maschinenwortes
UP	Unterprogramm
UUP	Unter-Unterprogramm
V	Vergleicher
Vz	Vorzeichen
WR	Wagenrücklauf
WRZL	Wagenrücklauf-Zeilenvorschub
Z	Operationszusatz
z	Zahl
z_{0-32}	Dualzahl 0-32 des Maschinenwortes
ZWR	Zwischenraum
:=	ergibt sich aus
> <	Adresse
< >	Inhalt

Ag 117-III-D-1-83- 0338